UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Modelos de predicción de la demanda de energía eléctrica a corto plazo para pequeños grupos de consumidores

Autor/es

RAÚL FERNÁNDEZ RUIZ

Director/es

LUIS ALFREDO FERNÁNDEZ JIMÉNEZ

Facultad

Escuela Técnica Superior de Ingeniería Industrial

Titulación

Grado en Ingeniería Eléctrica

Departamento

INGENIERÍA ELÉCTRICA

Curso académico

2018-19

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL

TRABAJO DE FIN DE GRADO

TITULACIÓN: Grado en Ingeniería Eléctrica

CURSO: **2018/2019** **CONVOCATORIA: JULIO**

TÍTULO:

Desarrollo de modelos de predicción a corto plazo de la demanda de energía eléctrica para pequeños grupos de consumidores

ESTUDIANTE: Raúl Fernández Ruiz

TUTORES/AS: Luis Alfredo Fernández Jiménez

DEPARTAMENTO: Ingeniería Eléctrica

1. Índices

1.1 Índice de capítulos

1.2 Índice de figuras

1.3 Índice de tablas

2. Abreviaturas

AIC	Criterio de información de Akaike
CT	Centro de transformación
GFFN	Generalized feed forward networks (redes de avance generalizadas)
MAE	Mean absolute error (Error medio absoluto)
MAPE	Mean absolute percentage error (Error medio absoluto porcentual)
MLP	Multilayer perceptron (Perceptrón multicapa)
RBF	Radial Basis Function (Funciones de base radial)
RNA	Redes neuronales artificiales
RMSE	Root mean square error (Raiz del error cuadrático medio)
SVM	Support vector machines (Máquinas de vector soporte)
VE	Variables exógenas

3. Resumen

En el presente estudio se elaborarán diferentes modelos predictivos aplicados a la demanda de energía eléctrica para un pequeño grupo de consumidores. Tomando como datos de partida la potencia demandada de un centro de transformación y las predicciones meteorológicas de la ciudad de Logroño, se generarán diversas variables explicativas del término de demanda de potencia eléctrica.

Tras realizar una selección de las variables para utilizar únicamente las que presenten una mayor relación con el término de potencia a predecir la muestra de datos se dividirá en tres partes: una utilizada para elaborar y entrenar los modelos, otra para determinar cuándo dejar de entrenarlos y una tercera para evaluar los resultados.

Se utilizarán diferentes programas informáticos para generar los modelos basados en regresión lineal, redes neuronales, inteligencia difusa y bosques aleatorios compuestos por árboles de regresión.

La comparativa de resultados se realizará, en primer lugar, únicamente entre los modelos realizados con el mismo programa informático y a continuación de modo global, para determinar cuál de todos es el mejor modelo.

4. Abstract

The following research will develop different predictive models applied to electricity demand for a small group of consumers. Taking as starting data the demanded power of a low voltaje substation and the weather forecast of Logroño city.

After making a selection of the variables in order to use only those that have a greater relationship with the power term to predict the data sample will be divided into three parts: the first one used to develop and train the models, the second one to determine when to stop training and the last one to evaluate the results

Diferent computer programs will be used to generate models based on inear regression, neural networks, diffuse intelligence and random forests composed of regression trees,

The comparison of results will be made. firstly only between the models developed with the same computer program and then globally to determine which of them is the most appropriate

5. Introducción

Tras los grandes avances en energías renovables, generación distribuida e inclusión del coche eléctrico en la sociedad actual, son necesarios los modelos de predicción de la demanda integrados en una red inteligente (*smart grid* en inglés) para una adecuada gestión energética.

El mercado eléctrico, en los próximos años, va a experimentar un notable cambio a costa de la progresiva desaparición de las centrales nucleares y térmicas de fuel y carbón, las cuales serán sustituidas por energías renovables, en su mayor parte eólica y solar fotovoltaica. Este cambio supondrá la transición de unas energías cuya generación es controlada a otras en las que la generación depende de la climatología, por lo que cada día se va a tender más a la creación de una red inteligente de gestión de la energía.

Mediante la generación distribuida y el autoconsumo se pretende generar la energía, a diferencia del modelo de generación concentrada actual, cerca de los puntos de consumo, minimizando pérdidas de transmisión en la red y descongestionando ésta, lo que implicará, que por ejemplo, en el caso de que un usuario que disponga de un autoconsumo conectado a red solo demandará la energía que ese autoconsumo no sea capaz de generar, lo que hará que su curva de carga tenga se vea influenciada en gran medida por la generación, necesitándose de un modelo de previsión de demanda que contemple estos aspectos.

Esta previsión de la demanda puede permitir, por ejemplo, para una gran empresa con una autogeneración mediante renovables, la posibilidad de conocer la demanda neta energética que se producirá en un momento dado, fruto de la diferencia entre la demanda real y la generación mediante fuentes de energía de origen renovable. También podría ser de gran interés para una comercializadora, con centros de transformación distribuidos en una ciudad, conocer la demanda futura de energía, de cara a compras en los mercados diario, intradiario o a futuros. Otro caso con especial interés podría ser el de, uno varios pequeños consumidores, con disponibilidad de un vehículo eléctrico y conectados a una red inteligente de carga, siendo el vehículo eléctrico de cada uno de ellos un elemento más del sistema, usado como fuente de acumulación de energía, pudiendo estar al servicio de las necesidades de esa red haciendo más posible la gestión de la energía solar fotovoltaica y eólica, entre otras renovables cuya producción no es controlable.

Es por todo esto, por lo que el desarrollo de nuevos modelos de predicción de la demanda se hacen indispensables, de cara a conocer, en tiempo real la estimación de demanda futura para un periodo de tiempo en base a la época del año, demanda en días previos, previsiones meteorológicas etc.

6. Objetivos

Con el presente Trabajo Fin de Grado se pretende como objetivo principal el de desarrollar el mejor modelo de predicción, a corto plazo, de la demanda de un conjunto de pequeños consumidores. En revistas científicas especializadas se han descrito, en las tres últimas décadas, multitud de modelos de predicción de la demanda, pero se han aplicado a nivel nacional o regional, siendo los trabajos dirigidos al desarrollo de modelos de predicción de la demanda de pequeños grupos de consumidores prácticamente inexistentes.

Como objetivos secundarios se han fijado los siguientes:

- Conocer y analizar diferentes modelos matemáticos para la predicción de la demanda.
- Determinar cuáles de las posibles variables explicativas del término de potencia son más relevantes.
- Realizar una comparativa entre todos los modelos estudiados a fin de determinar cuál se comporta mejor.
- Analizar los errores producidos por los modelos de predicción y determinar las técnicas que permitan su minimización.

7. Desarrollo del trabajo

7.1 Elaboración de la muestra

Para la elaboración de los modelos de predicción de la demanda, se han utilizado dos bases de datos:

- Potencias demandadas en un centro de transformación (en lo sucesivo CT) ubicado en la ciudad de Logroño entre los años 2008 y 2013.
- Predicciones meteorológicas para la ciudad de Logroño con 1 día de antelación entre los años 2008 y 2013.

7.1.1 Potencias demandadas

Para la elaboración de modelos matemáticos para la predicción de la demanda se ha utilizado como punto de partida los valores de potencia de consumo de un CT de la ciudad de Logroño desde el 1 de enero de 2008 hasta el 8 de febrero de 2013, lo que supondrá un total de 5 años 1 mes y 8 días.

Estos valores fueron registrados con una periodicidad de 5 minutos; siendo el valor registrado, por ejemplo, a las 00:00 el día D, el valor de los 5 minutos anteriores a la hora fijada, es decir desde las 23:55 del día D-1 hasta las 00:00 del día D. Por ello para obtener los valores de demanda de potencia hora a hora se deberá sumar, para cada hora, los valores desde el minuto 5 hasta el minuto 0 correspondiente a la hora siguiente.

En primer lugar, se realizará una representación de las potencias para evaluar posibles faltas de datos u otros posibles defectos de la muestra, como se puede apreciar en la *figura 7.1*.

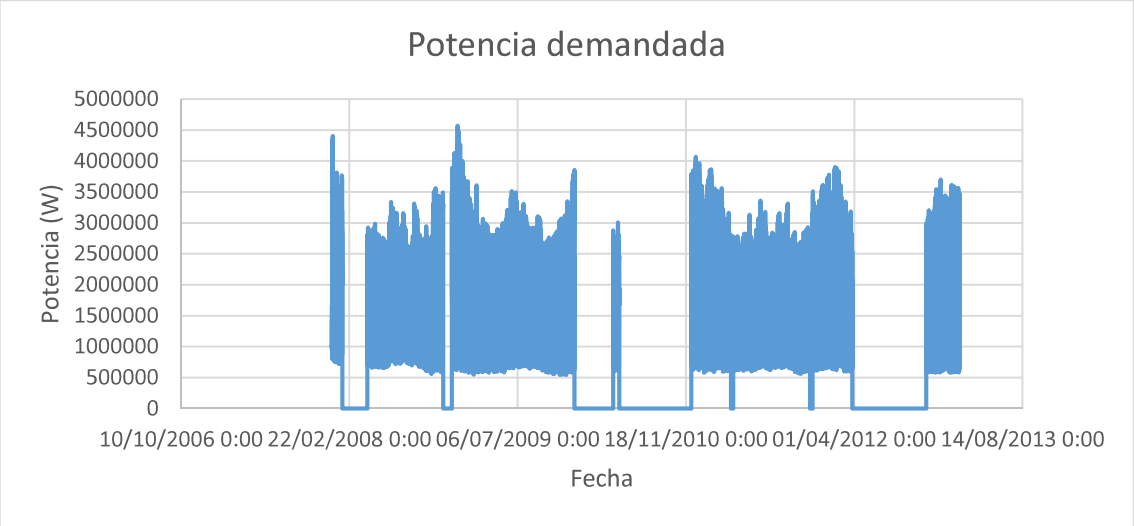

Figura 7.1: Potencia demandada en el periodo de la muestra

De la figura 7.1 se deduce que hay periodos de tiempo en los que la medida de potencias no se realizó, suponiendo esto el primer problema, haciendo que la muestra no tenga la cantidad de datos supuestos inicialmente.

Partiendo de estos datos de potencias se han creado otras posibles variables explicativas como son las siguientes:

- Potencia demandada 24 horas antes (P-24).
- Potencia demandada 48 horas antes (P-48).
- Potencia demandada 168 horas antes (P-168).
- Día de la semana: número del día de la semana siendo el domingo el día 1 y el sábado el día 7.
- Sábado: la variable presentará un 1 si el día coincide con el citado y un 0 si no es así.
- Domingo: la variable presentará un 1 si el día coincide con el citado y un 0 si no es así.
- Lunes: la variable presentará un 1 si de la semana es festivo nacional y un 0 si no es así.
- Festivo: la variable presentará un 1 si el día coincide con el citado y un 0 si no es así.
- Hora: Número de hora del 0 al 23 correspondiente al horario GMT+1.

7.1.2 Predicciones meteorológicas

Se ha considerado que la potencia demandada por el CT estudiado puede tener cierta relación con la meteorología. Esta suposición responde a la relación que puede tener el consumo eléctrico con la temperatura exterior, con la radiación, con la humedad relativa, etc. Por ello, se han tomado ciertas variables meteorológicas como posibles variables predictivas o explicativas de la potencia demandada.

Del mismo modo que los valores de consumos del CT, se dispone de las predicciones meteorológicas para la ciudad de Logroño desde el 1 de enero de 2008 hasta el 8 de febrero de 2013 realizadas con las siguientes antelaciones: 1 hora, 1 día, 2 días y 3 días. Las predicciones se han obtenido del servidor del Servicio Meteorológico de Galicia (MeteoGalicia), interpolando los cuatro puntos de la rejilla de análisis más próximo a la ciudad de Logroño.

Para la elaboración de los modelos de predicción de la demanda se han escogido las previsiones realizadas con 1 día de antelación. De igual modo que con los datos de potencias consumidas, en las predicciones existen periodos de tiempo sin datos, los cuales han sido rellenados con las predicciones realizadas con 2 y 3 días de anterioridad de las cuales si se tenía datos.

Las variables meteorológicas que se han tenido en cuenta para la elaboración de los modelos han sido las siguientes:

- Temperatura.
- Radiación.
- Velocidad de viento.
- Humedad.
- Fracción de nubes.
- Precipitación.
- Calor latente.
- Calor sensible.

7.2 Tipos de modelos para la predicción de la demanda

La organización de los siguientes capítulos vendrá definida por el programa informático mediante el cual se realizarán los modelos de predicción de la demanda siguiendo el siguiente orden:

1. **Modelos creados mediante Excel:** Modelos de regresión lineal.
2. **Modelos creados mediante el programa informático Neurosolutios.** Modelos basados en redes neuronales artificiales (en lo sucesivo RNA)
3. **Modelos creados mediante el programa informático Matlab.** Modelos basados en lógica difusa.
4. **Modelos creados mediante el programa informático RStudio.** Modelos de tipo regresión lineal, redes neuronales y bosques aleatorios.

7.3 Modelos elaborados mediante el programa informático Excel

7.3.1 Consideraciones para el estudio con modelos de tipo lineal mediante Excel

Dadas las limitaciones del programa informático Excel para la realización de modelos de predicción únicamente se realizarán dos tipos de modelos:

1. **Modelos de días previos:** Estos modelos se basan en que el consumo de un día futuro será el mismo que el consumo de un día previo considerado. Son modelos muy simples que, como se verá en capítulos posteriores, no se ajustan demasiado bien a la temporalidad.

2. **Modelos de regresión lineal**: Estos modelos son modelos matemáticos basados en combinaciones lineales de variables explicativas usadas para aproximar una variable dependiente, que será la potencia demandada y tendrán la siguiente forma:

$$P_{hora\ h} = a_0 + a_1 \cdot X_1 + a_2 \cdot X_2 + \cdots + a_n \cdot X_n \qquad (7.1)$$

Siendo:

$P_{hora\ h}$ = Potencia prevista a consumir en el día D a la hora h.

$a_0, a_1, \ldots, a_n$: Parámetros que miden la influencia de las variables explicativas

$X_1, X_2, \ldots, X_n$: Variables explicativas o independientes

7.3.1.1 Subdivisión de la muestra

En este primer tipo de modelos se ha subdividido la muestra en dos partes notablemente diferenciadas:

1. **Muestra de entrenamiento:** Se tomó esta parte de la muestra como datos con los que entrenar y optimizar el modelo de modo que este adquiera el comportamiento deseado. Se utilizaron los primeros 20477 periodos horarios, es decir, un total de 853 días aproximadamente, incluyendo en ellos posibles faltas de datos.
2. **Muestra de evaluación:** Esta parte de la muestra fue utilizada para, mediante el modelo elaborado con la muestra de entrenamiento, comprobar los resultados que con él se obtienen. Esta parte de la muestra se compone de los datos entre los periodos 25623 hasta 44777 (no se utilizan los datos de los periodos entre el 20478 y el 25622 dado que no se dispone de medida en esos periodos), lo que hace un total de 19154 periodos de 1 hora, lo que hace un total de 798 días aproximadamente, incluyendo en ellos, al igual que antes, las posibles faltas de datos.

7.3.1.2 Tratamiento de datos

- Para ningún modelo es posible realizar predicciones y compararlas de los periodos temporales de los que no se tienen datos de consumo.
- Para los modelos basados en potencias demandadas N días antes se deberá disponer de esos datos de demanda, lo que significa, por ejemplo, que para un modelo basado en la potencia del día D-6 para la predicción de la

potencia D+1, no será posible elaborar una predicción hasta 7 días después del restablecimiento de toma de valores tras una falta.

- Los modelos se optimizarán con Excel, mediante la función solver, estableciendo como objetivo la minimización del error porcentual absoluto medio (en lo sucesivo MAPE), contemplando también el error cuadrático medio (root mean square error (RMSE)) y el error absoluto medio (MAE) cuyas fórmulas son las siguientes:

$$MAPE(\%) = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{|P_i - P_i^*|}{P_i} \cdot 100 \quad (7.2)$$

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |P_i - P_i^*| \quad (7.3)$$

$$RMSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (P_i - P_i^*)^2 \quad (7.4)$$

Siendo:

P_i: *Potencia* real.

P_i^*: Potencia predicha.

n: El número de datos

- El error porcentual absoluto medio y el error cuadrático medio se evaluarán tanto en la muestra de entrenamiento como en la de evaluación, a fin de comparar estos errores en ambas muestras y determinar si el modelo está sobreentrenado o muy ajustado a la muestra de entrenamiento, o, en caso contrario, es un modelo generalista válido para cualquier muestra de datos.

7.3.2 Tipos de modelos lineales con Excel

7.3.2.1 Modelo de día anterior

Para la elaboración de este modelo, se tiene en cuenta, que la potencia consumida el día D a la hora h, es igual a la potencia consumida el día D-1 a la hora h.

Esta estimación se basa en que la potencia consumida dos días consecutivos debe ser similar.

Elegido un periodo arbitrario de 25 días el error porcentual absoluto medio es el representado en la *figura 7.2.*

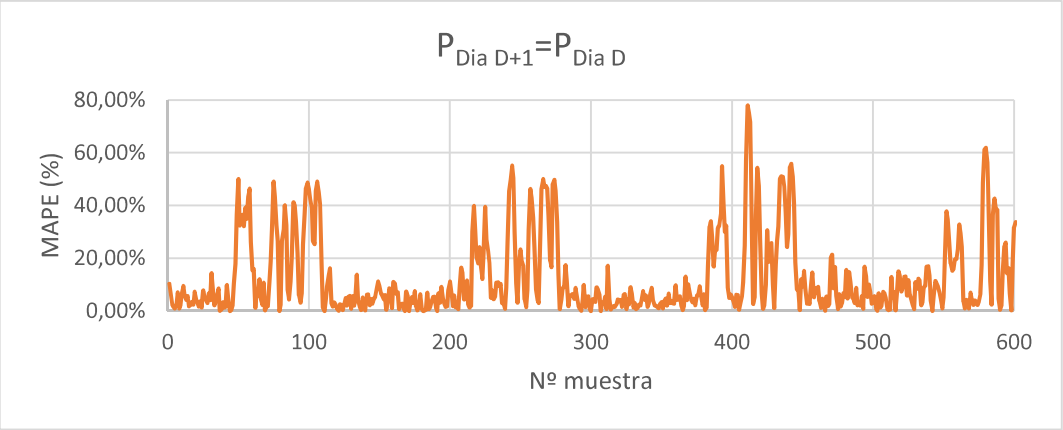

Figura 7.2: Error MAPE para un periodo de 600 horas de la muestra de evaluación con el modelo del día anterior.

En la *figura 7.2* puede observarse cierta periodicidad en el error, debido a que las predicciones, por ejemplo, del lunes son basadas en el domingo, siendo ambos días totalmente diferentes en cuanto a consumos. En base a este resultado cabe esperar que el modelo basado en un día similar debe presentar un mejor ajuste.

7.3.2.2 Modelo de día similar

Se utiliza el valor de la potencia hora a hora consumida el día D-6 como predicción de la potencia a consumir el día D+1.

Este modelo, tiene en cuenta los fallos aparentes en el modelo del día anterior, que no contempla que la potencia consumida en días consecutivos puede ser muy diferente.

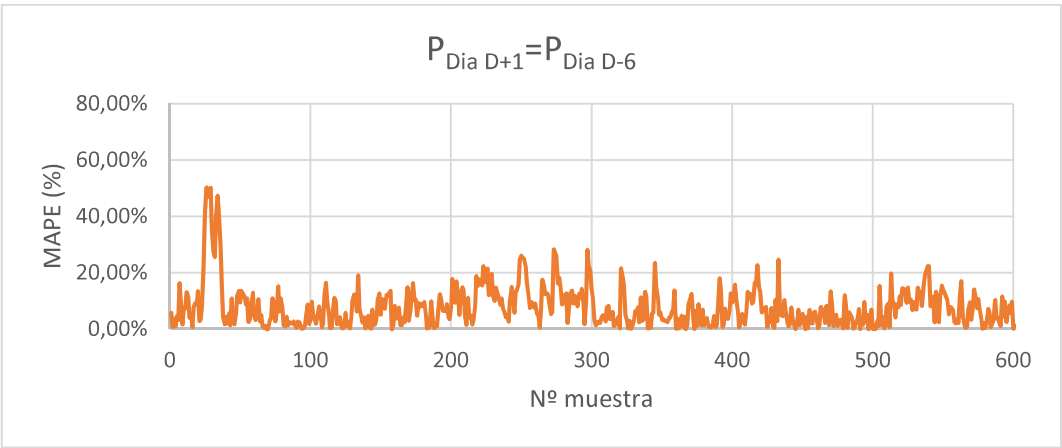

Figura 7.3: Error MAPE para un periodo de 600 horas de la muestra de evaluación con el modelo de día similar

Como era de esperar, en la *figura 7.3* puede apreciarse que el resultado obtenido con el modelo anterior se ve notablemente mejorado con respecto al del modelo de día anterior.

Aun siendo este modelo mejor que el anterior surge la duda sobre si, una combinación lineal de ambos, junto a otras variables mejorarían la respuesta del sistema dando lugar a los modelos regresivos.

7.3.2.3 Modelos regresivos basados en potencias de días previos

En este apartado se evaluarán diferentes variantes de modelos regresivos utilizando como variables las potencias consumidas a la misma hora en días anteriores.

Variante 1

$$P_{día\ D+1} = a_0 + a_1 \cdot P_{día\ D} \tag{7.5}$$

El valor de la potencia predicha hora a hora se compondrá de un valor de potencia fijo a_0 sumado a un término fijo a_1 multiplicado por la potencia del día anterior.

Este modelo se apoya en que la potencia del día D+1 tiene una relación con la potencia del día D.

Una vez optimizado, utilizando la hoja de cálculo Excel y sobre las muestras de entrenamiento, el modelo responde a la *ecuación 7.6*.

$$P_{día\ D+1} = 6{,}042 \cdot 10^{-7} + 0{,}97826 \cdot P_{día\ D} \tag{7.6}$$

Variante 2

$$P_{día\ D+1} = a_0 + a_1 \cdot P_{día\ D} + a_2 \cdot P_{dia\ D-1} \tag{7.7}$$

El valor de la potencia predicha hora a hora se compondrá, además de un término fijo a_0, de dos términos fijos, a_1 y a_2 multiplicados por la potencia de los 2 días anteriores al día a predecir.

Este modelo, contempla que la potencia del día D+1 a la hora h tiene relación con la potencia consumida de los dos días anteriores a esa misma hora h.

Una vez optimizado, utilizando la hoja de cálculo Excel y sobre las muestras de entrenamiento, el modelo responde a la *ecuación 7.8*.

$$P_{día\ D+1} = 5{,}1267 \cdot 10^{-5} + 0{,}95875 \cdot P_{día\ D} + 0{,}02026 \cdot P_{dia\ D-1} \tag{7.8}$$

Variante 3

$$P_{día\ D+1} = a_0 + a_1 \cdot P_{día\ D} + a_2 \cdot P_{dia\ D-1} + a_3 \cdot P_{día-6} \quad (7.9)$$

En este modelo, el valor de la potencia predicha hora a hora tiene en cuenta, los mismos términos que el modelo anterior sumado a un término fijo a_3 multiplicado por la potencia consumida a la misma hora el día D-6.

Este modelo, contempla que la potencia el día D+1 a la hora h tiene relación con la potencia consumida el día anterior, dos días antes y 7 días antes a la misma hora h.

Una vez optimizado, utilizando la hoja de cálculo Excel y sobre las muestras de entrenamiento, el modelo responde a la *ecuación 7.10*

$$P_{día\ D+1} = 5{,}12665 \cdot 10^{-7} + 0{,}20459 \cdot P_{día\ D} + 0 \cdot P_{dia\ D-1} + 0{,}78003 \cdot P_{día\ D-6} \quad (7.10)$$

Variante 4

$$P_{día\ D+1} = a_0 + a_1 \cdot P_{día\ D} + a_2 \cdot P_{dia\ D-1} + a_3 \cdot P_{día\ D-6} + a_4 \cdot P_{min} \quad (7.11)$$

Con este modelo se intenta implementar el efecto que tiene la potencia mínima del día anterior en la potencia a predecir en el día siguiente mediante un término a_4 multiplicado por la potencia mínima del día anterior.

Una vez optimizado, utilizando la hoja de cálculo Excel y sobre las muestras de entrenamiento, el modelo responde a la *ecuación 7.12.*

$$P_{día\ D+1} = 5{,}127 \cdot 10^{-7} + 0{,}19984 \cdot P_{día\ D} + 0 \cdot P_{dia\ D-1} + 0{,}7758 \cdot P_{día-6} + 0{,}01775 \cdot P_{min} \quad (7.12)$$

7.3.2.4 Modelos regresivos basados en potencias de días previos y otras variables

Variante 5: Modelos regresivos basados en potencias de días previos y variables exógenas

$$P_{día\ D+1} = a_0 + a_1 \cdot P_{día\ D} + a_2 \cdot P_{dia\ D-1} + a_3 \cdot P_{díaD-6} + a_4 \cdot P_{min} + a_5 \cdot Temperatura \quad (7.13)$$

En este modelo, mediante la adicción de una variable adicional, como es la temperatura prevista, multiplicada por un término fijo a_6 se ha intentado contemplar el efecto de la temperatura exterior en la demanda de potencia.

Una vez optimizado, utilizando la hoja de cálculo Excel y sobre las muestras de entrenamiento, el modelo responde a la siguiente ecuación:

$$P_{día\ D+1} = 5{,}127 \cdot 10^{-7} + 0{,}19913 \cdot P_{día\ D} + 0 \cdot P_{dia\ D-1} + 0{,}77587 \cdot P_{día-6} + 0{,}00114 \cdot P_{min} + 41{,}7035 \cdot Temperatura \quad (7.14)$$

Variante 6: Modelos regresivos basados en potencias de días previos, variables exógenas y variables ficticias

$$P_{día\ D+1} = a_0 + a_1 \cdot P_{día\ D} + a_2 \cdot P_{dia\ D-1} + a_3 \cdot P_{día\ D-6} + a_4 \cdot P_{min} + a_5 \cdot Temperatura + a_6 \cdot Sábado + a_7 \cdot Domingo + a_8 \cdot Festivo + a_9 \cdot Lunes \quad (7.15)$$

Por último, en este modelo, se han implementado 4 variables binarias adicionales (variables "dummy") correspondientes a los días: sábado, domingo, lunes y festivo multiplicadas por 4 términos fijos a_6, a_7, a_8 y a_9 para reflejar como afecta la estacionalidad en la demanda de potencias.

Una vez optimizado, utilizando la hoja de cálculo Excel y sobre las muestras de entrenamiento, el modelo responde a la siguiente ecuación:

$$P_{día\ D+1} = 5{,}127 \cdot 10^{-7} + 0{,}19922 \cdot P_{día\ D} + 0 \cdot P_{dia\ D-1} + 0{,}7758 \cdot P_{día\ D-6} + 0{,}00091 \cdot P_{min} + 42{,}0841 \cdot Temperatura + 35{,}5652 \cdot Sábado + 0 \cdot Domingo + 0 \cdot Festivo + 155{,}544 \cdot Lunes \quad (7.16)$$

7.4 Modelos elaborados mediante Neurosolutions

Dado que el programa informático Neurosolutions es un programa para la realización de modelos de predicción mediante redes neuronales, en los sucesivos apartados se introducirán los conceptos clave de este tipo de redes, su modo de funcionamiento y la metodología utilizada para la división de la muestra de datos para su tratamiento.

7.4.1 Marco teórico

7.4.1.1 La neurona biológica y el proceso de aprendizaje humano

El sistema nervioso humano tiene dos partes claramente diferenciadas, el sistema nervioso central, compuesto por el cerebro y la médula espinal y el sistema nervioso periférico.

La neurona biológica es el componente principal del cerebro. Puede ser definida como un tipo especializado de célula que integra una entrada, conectada a otras neuronas y propaga a través de una salida la señal hacia más neuronas, esta secuencia se realiza gracias a una serie compleja de eventos bioquímicos. Sus partes principales son:

Dendritas: Son protuberancias del cuerpo celular encargadas de recoger señales químicas de otras neuronas.

Cuerpo celular: Es el encargado de integrar las señales eléctricas de todas las dendritas y convertirlas en una serie de potenciales eléctricos.

Axón: Proyección larga y delgada a través de la cual se propagan las señales hacia otras neuronas.

Terminales: Extremos de ramificación del axón en los que la actividad eléctrica se convierte en señal bioquímica con la que estimular otra neurona.

Sinapsis: Pequeña unión entre los terminales de una neurona y las dendritas de otra, es donde la señal bioquímica de una neurona se convierte en señal eléctrica en la siguiente.

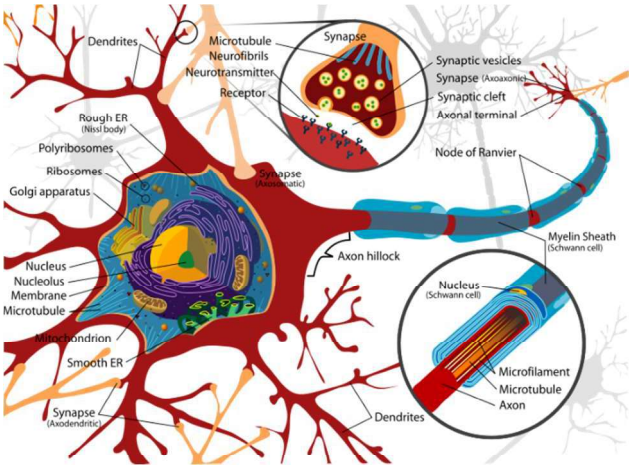

Figura 7.4 Diagrama de una neurona biológica.
Recuperado de (Ruiz, 2007)

7.4.1.2 Introducción a las RNA

Las RNA son un modelo computacional inspirado en el comportamiento de una red neuronal biológica.

La estructura más básica de las RNA presenta una capa de entrada, en la que habrá tantas neuronas de entrada como variables explicativas se consideren y una capa de neuronas de salida, estando ambas interconectadas mediante unos enlaces de conexión o enlaces sinápticos. En modelos de RNA más avanzados podrán existir una o varias capas ocultas de neuronas intermedias con tantas neuronas como se quiera por capa.

7.4.1.3 Neuronas

La unidad principal de una RNA es la neurona, esta es una unidad de procesamiento de información con tres elementos fundamentales:

1- **Enlaces de conexión:** Parten de unas neuronas y llegan a parar a otras, cada uno de ellos está ponderado con un peso propio. El valor x_j, procedente de la capa de neuronas inmediatamente anterior, es multiplicado por el peso sináptico w_{kj} donde el primer subíndice indica la neurona en cuestión y el segundo subíndice indica a que entrada corresponde el peso en cuestión.

2- **Función de ponderación:** esta función se ocupa de convertir el resultado de todos los productos de los enlaces de conexión por las respectivas entradas en un único valor. Existen fundamentalmente dos funciones de ponderación:

- Suma ponderada de las entradas multiplicadas por sus pesos

$$v_k = \sum_{j=1}^{m} w_{kj} \cdot x_j + b_k \qquad (7.17)$$

- Distancia euclídea: Es utilizada en redes no supervisadas de modo que la red se ajuste a los patrones.

$$v_k = \sum_{j=1}^{m} \left(w_{kj} \cdot x_j\right)^2 + b_k \qquad (7.18)$$

Donde $x_1, x_2, \ldots, x_m$ son términos de entrada; $w_{k1}, w_{k2}, \ldots, w_{km}$ son pesos sinápticos de la neurona k y u_k es la función de ponderación.

3- **Función de activación:** Se ocupa de limitar la salida de la neurona, establece un umbral que se debe sobrepasar para que la información pueda

propagarse a la siguiente neurona, en determinados casos hace que la salida de la neurona tome valores finitos.

$$y_k = \varphi(v_k) \tag{7.19}$$

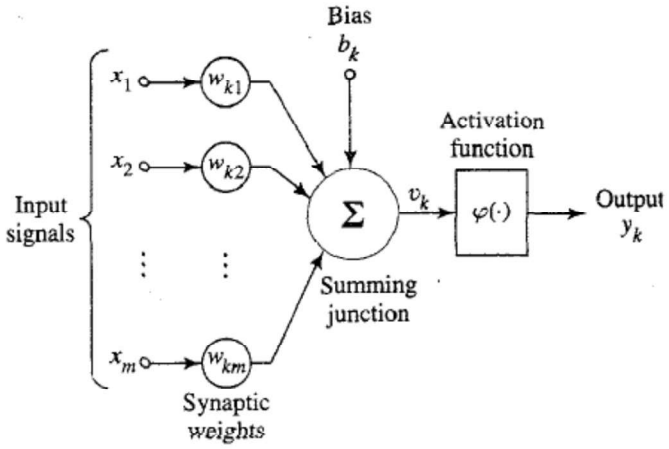

Figura 7.5: Estructura de una neurona

Recuperado de (Haykin, 1999)

En el modelo de la *figura 7.5* aparece el término b_k (bias) que incluye un término aplicado externamente incrementando o decrementando la entrada a la función de activación.

7.4.1.4 Tipos de función de activación

Anteriormente se ha hablado de la función de activación, esta es la función de cada neurona que convierte las entradas ponderadas de una neurona a su salida. Se identificarán 3 tipos básicos de función de activación:

1. Función umbral, para los diferentes valores de v se tiene:

$$\varphi(v) = \begin{cases} 1 \; si \; v \geq 0 \\ 0 \; si \; v \leq 0 \end{cases} \tag{7.20}$$

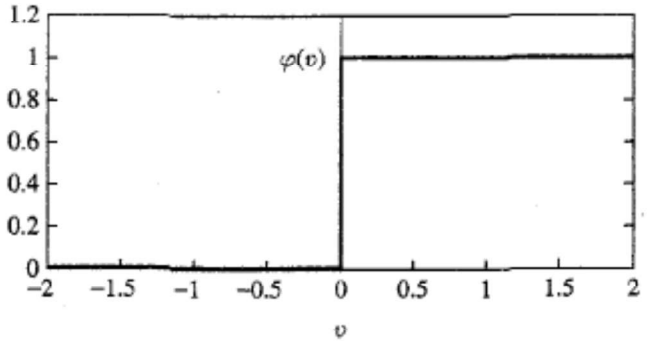

Figura 7.6: Representación de la función de activación umbral

Recuperado de (Haykin 1999)

2. Función a trozos, para los diferentes valores de *v* se tiene:

$$\varphi(v) = \begin{cases} 1, & v \geq 0{,}5 \\ v + 0{,}5, & 0{,}5 > v > -0{,}5 \\ 0, & v \leq -0{,}5 \end{cases} \tag{7.21}$$

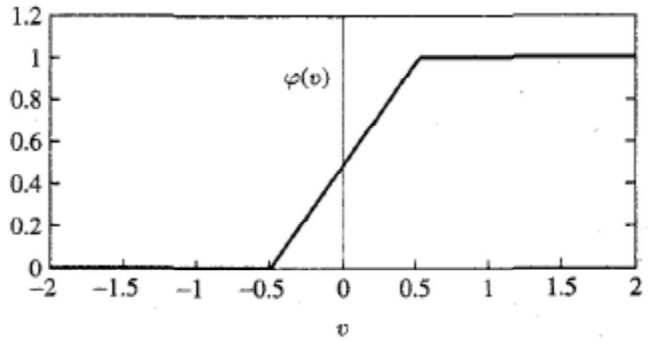

Figura 7.7: Representación de la función de activación a trozos

Recuperado de (Haykin 1999)

3. Función sigmoidea: Es una de las funciones de activación más comunes junto con la tangente hiperbólica y tiene la forma:

$$\varphi(v) = \frac{1}{1 + \exp(av)} \tag{7.22}$$

Donde *a* es un parámetro de la función sigmoidea, variando el parámetro *a* se obtienen funciones sigmoideas con diferentes pendientes como se ilustra en la *figura 7.8*, de este modo la función sigmoidea hace que la función de activación pueda tomar cualquier parámetro entre 0 y 1.

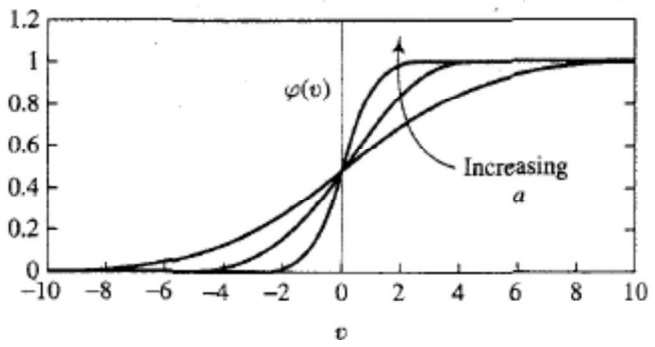

Figura 7.8: Representación de la función de activación sigmoidea

Recuperado de (Haykin 1999)

Las funciones explicadas anteriormente toman valores entre 0 y 1, pero a veces, es deseable que los valores de la función de activación estén entre -1 y 1, tomando la función de activación una forma antisimétrica respecto del origen, la función de la *ec. 5.23* quedará del siguiente modo:

$$\varphi(v) = \begin{cases} 1 & si \quad v \geq 0 \\ 0 & si \quad v = 0 \\ -1 & si \quad v < 0 \end{cases} \tag{7.23}$$

4. Función tangente hiperbólica: Si lo que se desea es que la función de activación presente valores continuos entre -1 y 1 se requiere a la función tangente hiperbólica definida por:

$$\varphi(v) = \tanh(v) \tag{7.24}$$

Tanto la tangente hiperbólica como la función sigmoide hacen que un pequeño valor de cambio en la entrada produzca un pequeño cambio en la salida lo que hace muy interesante su uso.

7.4.1.5 Arquitectura de redes

En general se pueden identificar tres tipos fundamentales de RNA:

1. Redes de avance de una sola capa

Las RNA se organizan en capas, la organización más simple la componen una capa de entradas también llamadas fuentes, que contienen las posibles variables explicativas, que se conectan con una capa de neuronas posterior que es la capa de salidas, en la que se obtendrán los valores objetivo. Esta red, pese a tener dos capas, se denomina red de avance de una sola capa ya que no se cuenta la red de entradas en las que no se efectúa ninguna transformación.

2. Redes de avance de múltiples capas

Estas RNA se caracterizan por la presencia de una o más capas de neuronas ocultas. Añadiendo más capas de neuronas ocultas la red la red es capaz de obtener resultados de mayor orden estadístico, lo que es evaluable cuando el número de variables en la capa de entrada es grande.

Las señales de los elementos de la capa de entradas son aplicadas a la segunda capa, y, la salida de las neuronas de la segunda capa es usada como entrada para la tercera capa. En definitiva, normalmente, las neuronas de cada capa de red tienen como entradas las salidas de las neuronas de la capa anterior. En la *figura 7.9* se representa una RNA de 10 nodos de entradas, una capa oculta con 4 neuronas y una capa de salidas con 2 neuronas de salida.

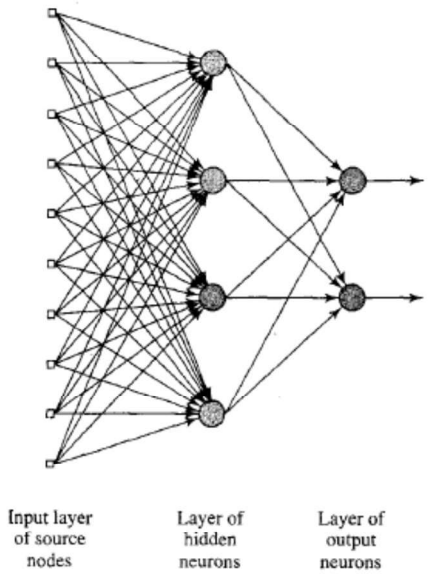

Figura 7.9: Red de avance de múltiples capas
Recuperado de (Haykin 1999)

3. Redes recurrentes

Esta red se distingue de las redes de avance hacia adelante en que tiene un lazo de realimentación. La presencia de lazos de realimentación tiene un profundo impacto en la capacidad de aprendizaje de la red, además, los lazos de realimentación suelen presentar dinámicas con comportamientos no lineales.

En la *figura 7.10* de representa una RNA de tipo recurrente con lazos de realimentación.

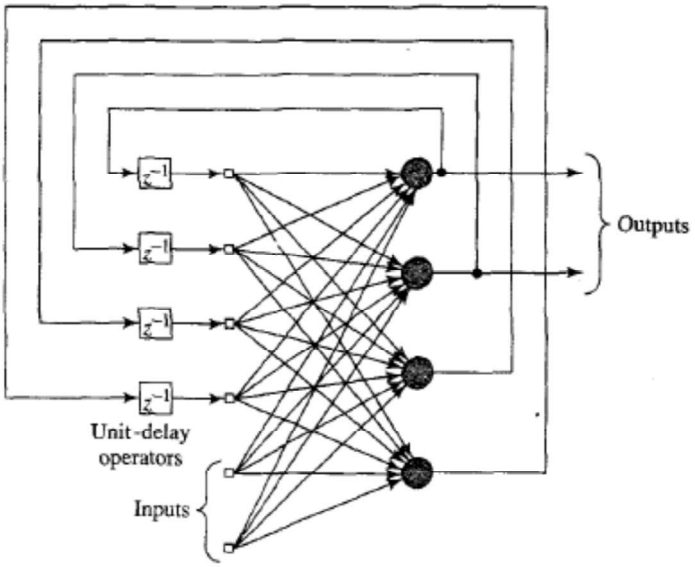

Figura 7.10: RNA recurrente

Recuperado de (Haykin 1999)

7.4.1.6 Proceso de aprendizaje

La propiedad que tiene una importancia más significativa en una RNA es la capacidad de esta de aprender sobre su entorno y mejorar su rendimiento conforme aprende. El aprendizaje de una RNA es un proceso mediante el cual los parámetros de una RNA se adaptan mientras el proceso de simulación se ejecuta. El tipo de aprendizaje es determinado por la manera en la que se producen cambios en los parámetros.

El proceso de aprendizaje implica la siguiente secuencia de eventos:

1. La RNA se simula en un entorno.
2. La RNA sufre cambios en sus parámetros como resultado de la simulación.
3. La RNA responde de un modo diferente al entorno debido a los cambios que han ocurrido en su estructura interna.

Los algoritmos de aprendizaje son un conjunto de reglas bien definidas para la resolución de un problema de aprendizaje, como es de esperar, no existe un único algoritmo de aprendizaje sino varios cada uno de los cuales ofrece unas ventajas propias.

7.4.1.7 Corrección de errores de aprendizaje

Para ilustrar la primera regla de aprendizaje se considerará una neurona *k* que constituye el único nodo computacional de la capa de salida de una RNA de avance hacia adelante. La neurona *k* es activada por una señal *x* creada por una o más capas de neuronas ocultas en base a un vector de entradas. La señal de salida de la neurona es el término y_k(n). Esta señal de salida representa la única salida de la RNA, que es comparada con la señal deseada denotada como d_k(n). Por lo que el error de la señal e_k(n) es:

$$e_k(n) = d_k(n) - y_k(n) \tag{7.25}$$

La señal de error e_k(n) actúa sobre el mecanismo de control, con el fin de aplicar una secuencia de ajustes correctivos sobre los pesos sinápticos de la neurona *k* con el fin de acercar la respuesta de la RNA a la respuesta deseada d_k(n) en un proceso iterativo.

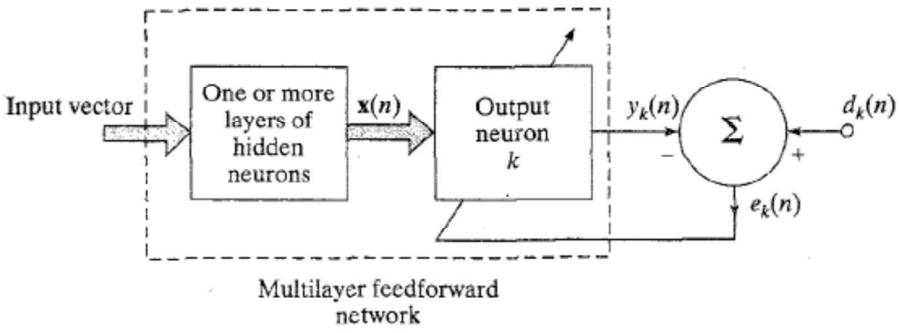

Figura 7.11: Mecanismo de corrección de errores de aprendizaje
Recuperado de (Haykin 1999)

El objetivo se centra en minimizar la función de coste del error ξ(n), que se define en términos del error e_k(n) como:

$$\varepsilon(n) = \frac{1}{2} \cdot e_k^2(n) \tag{7.26}$$

Esta función se define como valor instantáneo de la energía del error *ξ*(n). El proceso de ajuste de los pesos sinápticos de la neurona *k* se realiza reiteradamente hasta que el sistema llega a un estado estable.

El proceso de aprendizaje pretende reducir el valor de la energía del error mediante la regla delta o regla de Widrow- Hoff. El término ω_{kj}(n) denota el valor del peso sináptico

ω_{kj} de la neurona k excitada por el elemento x_j(n) en la iteración n. De acuerdo con la regla delta, el ajuste $\Delta\omega_{kj}$(n) es:

$$\boldsymbol{\Delta\omega_{kj}(n) = \eta \cdot e_k(n) \cdot x_j(n)} \quad (7.27)$$

Siendo η un término positivo constante que determina el ratio de aprendizaje que hay entre una iteración y la siguiente, estando este término entre 0 y 1.

Teniendo definido el ajuste sináptico $\Delta\omega_{kj}$(n), el valor actualizado de peso sináptico ω_{kj} es:

$$\omega_{kj}(n+1) = \omega_{kj}(n) + \Delta\omega_{kj}(n) \quad (7.28)$$

7.4.1.8 Tipos de aprendizaje

7.4.1.8.1 Aprendizaje con un maestro

En el aprendizaje con un maestro, también llamado aprendizaje supervisado, el maestro tiene conocimiento del entorno, conocimiento representado con un conjunto de pares entradas-salidas. En entorno es, sin embargo, desconocido para la RNA.

Suponiendo el caso de que la RNA y el maestro están expuestos a un vector de entrenamiento, el maestro puede proporcionar a la RNA la respuesta deseada para ese vector de entrenamiento, que es la respuesta óptima, dado que el maestro tiene conocimiento de ella. Esto hace que la RNA ajuste sus parámetros por la influencia combinada del vector de entrenamiento y del error, siendo el error la diferencia entre la señal deseada y la respuesta real de la RNA. Este ajuste se realiza de forma iterativa de modo que la RNA emule al maestro en sentido estadístico. De este modo el conocimiento del entorno por parte del maestro es totalmente transferido a la red para que esta mejore lo más posible. Cuando esto se consiga, entonces se podrá prescindir del maestro para que la RNA trabaje ella sola frente al entorno.

El aprendizaje supervisado se basa en el ya descrito método de corrección del error. Es un sistema de realimentación en lazo cerrado en el que el entorno no está dentro del lazo.

Como una medida de rendimiento se usa el error medio cuadrático (MSE) o la suma de los cuadrados de los errores sobre el vector de entrenamiento.

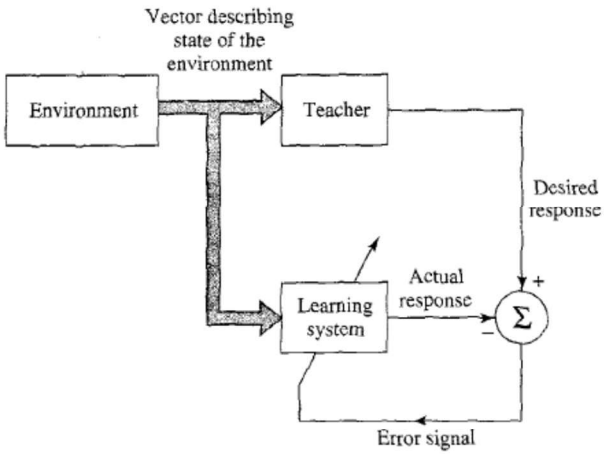

Figura 7.12: Representación del aprendizaje con maestro

Recuperado de (Haykin 1999)

Para que el sistema mejore su rendimiento a lo largo del tiempo, y, por lo tanto, disminuya el error debe tender hacia un mínimo, ya sea relativo o absoluto. Un sistema de aprendizaje supervisado utiliza el gradiente de la superficie del error correspondiente al comportamiento del sistema para mejorar su rendimiento, siendo el gradiente de la superficie el error un vector que señala la dirección en la que el error presenta un descenso más acusado. El uso de esta estimación puede tener una forma de caminata aleatoria pero mediante un algoritmo diseñado para minimizar la función de costo, un conjunto adecuado de entradas-salidas y el tiempo adecuado, un sistema de aprendizaje supervisado puede tener éxito en los ámbitos de determinación de patrones y aproximación de funciones.

7.4.1.8.2 Aprendizaje sin maestro

Este aprendizaje se realiza sin la supervisión de un maestro, lo que quiere decir que no hay ejemplos validados para que la red mejore. Existen dos subdivisiones en este aprendizaje:

7.4.1.8.3 Aprendizaje por refuerzo (programación neurodinámica).

El aprendizaje por refuerzo consiste en un mapeado de entradas-salidas de forma iterativa para minimizar un índice escalar de desempeño. En este aprendizaje se busca que la red de adapte de forma óptima al entorno, esto se consigue con la actuación de un crítico que se ocupa de premiar o castigar el comportamiento de la red en base a las acciones que esta adopte en base a que la red acierte o no en su respuesta, haciendo que esta transite a un estado diferente al que se encontraba.

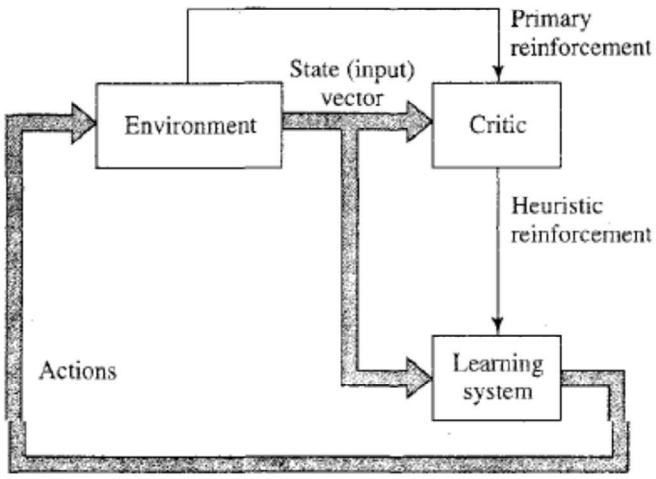

Figura 7.13: Esquema explicativo del aprendizaje por refuerzo

Recuperado de (Haykin 1999)

7.4.1.8.4 Aprendizaje no supervisado

En este tipo de aprendizaje no existe ningún ente externo que supervisa el proceso de aprendizaje como pueden ser el maestro o el crítico. La red tiene la capacidad de aprender los patrones de asociaciones para un conjunto de entradas en función de las características comunes de estas. Una vez que los patrones que rigen el comportamiento del sistema son aprendidos, la red es capaz de proporcionar una salida acorde al comportamiento de las entradas.

7.4.2 Modelos de RNA

7.4.2.1 Perceptrón simple (Single-Layer Perceptron)

Entre los años 1958 y 1962 el psicólogo Frank Ronsenblatt desarrolló un modelo simple de neurona con una regla de aprendizaje basada en la corrección del error a lo que llamó Perceptrón. Una de sus características principales es su capacidad de aprender a reconocer patrones. Este está constituido por un conjunto de sensores de entrada que reciben los patrones a reconocer y una neurona que se ocupa de clasificar a los patrones en caso de que su salida sea 0 o 1. Sus dos limitaciones más importantes son: la función de activación es de tipo escalón, por lo que su salida se limitará a una salida binaria y que no es capaz de aprender la función lógica XOR.

Se parte del modelo de neurona ya estudiado con anterioridad, el cual tiene una serie de entradas $x=(x_1,x_2,\ldots,x_n)$, unos pesos sinápticos de la neurona ω, $\omega(\omega_{k1},\omega_{k2},\ldots,\omega_{kn})$, un potencial sináptico $v_k=\sum(x_n\cdot\omega_{kn})$ y una función de activación.

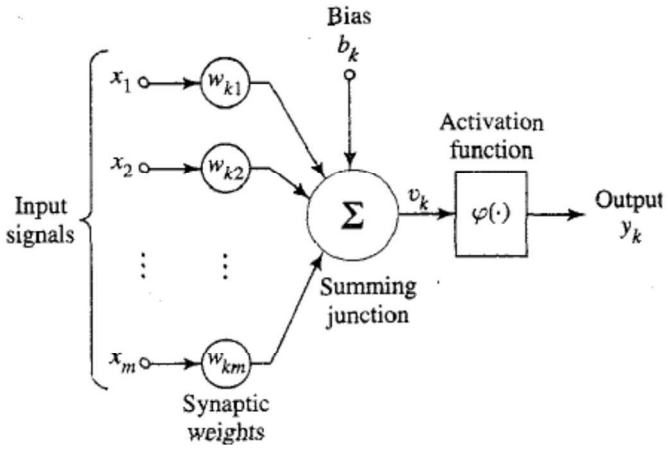

Figura 7.14: Estructura de una neurona

Recuperado de (Haykin 1999)

Para la determinación de los pesos sinápticos se parte de unos valores aleatorios de pesos sinápticos que se irán adaptando mientras que la salida y(n) no coincida con la salida deseada d(n). Esta regla se conoce como regla de aprendizaje del perceptrón simple y viene dada por la siguiente expresión:

$$w_j(n+1) = w_j(n) + \Delta w_j(n) \tag{7.29}$$

Siendo

$$\Delta w_j(n) = \eta(n) \cdot [d(n) - y(n)] \cdot x_j(n) \tag{7.30}$$

Lo que nos indica que la variación del peso w_j es proporcional al producto del error por la entrada x_j de la iteración k, es decir x_j(n) por el parámetro η(n) que representa el parámetro de la tasa de aprendizaje y está limitada a valores entre 0 y 1. Si el parámetro de tasa de aprendizaje es demasiado pequeño deriva en una velocidad de convergencia baja y la red puede quedar atrapada en un mínimo local, sin embargo, si el valor de la tasa de aprendizaje es alto puede derivar en inestabilidades en la función del error lo que hará que no se de una convergencia.

Por lo tanto, se utilizarán valores de tasas de aprendizaje altos al principio para ir disminuyendo progresivamente conforme la salida se acerque a la salida deseada.

7.4.2.2 Perceptrón multicapa (Multilayer Perceptron)

Estas redes consisten en una capa de entradas, una o más capas de neuronas ocultas y una capa de neuronas de salida. La señal se propaga a través de las capas hacia adelante, capa por capa. Estas redes son una generalización de las redes de una sola capa o de perceptrón simple y son denominadas redes de perceptrón multicapa (en lo sucesivo MLP).

Estas redes se basan en el algoritmo de propagación del error, este pasa dos veces a través de las neuronas de las diferentes capas, una vez hacia adelante y otra hacia atrás. En el paso hacia adelante, la señal de entrada es propagada desde la capa de entradas hasta la capa de neuronas de salida. Durante este primer paso los pesos sinápticos tienen un valor fijo. En el paso hacia atrás, los pesos sinápticos son ajustados en base a la regla de corrección del error o también llamado *algoritmo de propagación hacia atrás*, la señal de error es obtenida de la resta entre la señal objetivo y la señal generada por la RNA en sus neuronas de la capa de salidas. La regla de propagación del error se ocupa de ajustar los pesos de las conexiones sinápticas de modo que la respuesta de la red se acerque lo más posible a la respuesta deseada en sentido estadístico. La clave para este ajuste reside en que las neuronas de la capa oculta solo reciben una parte proporcional del error en función de su contribución a la neurona de salida.

Lo que puede ser expresado del modo siguiente:

$$\Delta\omega_{ji}(n) = -\eta \cdot \frac{\partial \xi(n)}{\partial \omega_{ji}} \tag{7.31}$$

siendo:

ξ(n): *valor instantáneo de la energía del error*, definido en términos del error $e_k(n)$ como:

$$\varepsilon(n) = \frac{1}{2} \cdot e_k^2(n) \tag{7.32}$$

η: *parámetro de tasa de aprendizaje*, que como se ha explicado en apartados anteriores es el responsable de la velocidad de convergencia de la salida.

El modelo de perceptrón multicapa tiene las siguientes características:

1. El modelo de cada neurona de la red incluye una función de activación de tipo no lineal, con una linealidad suave, lo que supone que un pequeño cambio en la entrada produce un pequeño cambio en la salida, a diferencia del

modelo de perceptrón de Rossenblatt que solo permitía respuestas binarias en la salida de las neuronas. Como función de activación suele utilizarse la función sigmoidea definida por la siguiente ecuación:

$$\varphi(v) = \frac{1}{1 + \exp(av)} \tag{7.33}$$

Donde a es un parámetro de la función sigmoidea, variando el parámetro a se obtienen funciones sigmoideas con diferentes pendientes como se ilustra en la *figura 7.15*, de este modo la función sigmoidea hace que la función de activación pueda tomar cualquier parámetro entre 0 y 1.

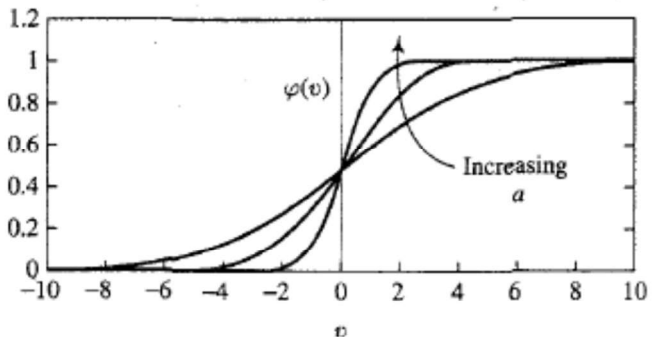

Figura 7.15: Variación de la función sigmoidea en función de a.
Recuperado de (Haykin 1999)

2. La RNA contienen una o más capas de neuronas ocultas que no son parte de las entradas ni de las salidas, cuya misión es capacitar a la red de aprender a realizar tareas complejas extrayendo características significativas de los parámetros de entrada.

7.4.2.3 Generalized Feed Forward Networks (GFFN)

Estas RNA únicamente presentan una diferencia con las RNA de tipo MLP y es que en las anteriores redes únicamente se permiten las uniones entre neuronas de capas consecutivas (p.e. de la capa de entradas a la primera capa de neuronas oculta), sin embargo, este tipo de red permite la conexión, mediante enlaces sinápticos entre neuronas de capas no consecutivas.

7.4.2.4 Funciones de base radial (RBF)

Las funciones de base radial (en lo sucesivo RBF) enfocan el trabajo de la RNA a un problema de ajuste de curvas en un espacio multidimensional. Por lo tanto el aprendizaje

es concebido como el proceso de encontrar una superficie en un espacio multidimensional que proporcione un mejor ajuste a los datos de entrenamiento.

Las redes de tipo RBF al igual que el resto de tipologías de redes tienen una capa de entradas y una capa de salidas, y varias capas ocultas, siendo la primera de ellas la capa multidimensional, y el resto capas de neuronas ocultas similares a las de las redes de perceptrón multicapa.

Para la confección de esta capa multidimensional de neuronas es utilizado un aprendizaje no supervisado, mediante el cual se crean agrupaciones (clusters), que son agrupaciones de los conjuntos de datos de entrenamiento que presentan un comportamiento similar. A continuación, se determinan las matrices de covarianza de los patrones de entrada de cada agrupación en función de la distancia euclídea entre cada punto y el centro de la agrupación. La función de activación de estas neuronas es una función gaussiana, haciendo que los valores cuya distancia euclídea se aleje más de los centros de las agrupaciones sean menos significativos que los más cercanos a los centros.

7.4.2.5 Análisis de componentes principales (PCA)

El análisis de componentes principales es una técnica que tiene como meta la reducción del tamaño de una muestra con la premisa de una pérdida de información mínima. Cada una de las componentes principales está creada a partir de las diferentes entradas mediante un método de aprendizaje no supervisado basado en una combinación lineal de las entradas.

La reducción del tamaño de la muestra se basa en eliminar cualquier redundancia a fin de crear unas variables más descriptivas de la solución, llamadas componentes principales (PCA), siendo estas variables más efectivas.

Tras la creación de las componentes principales, estas son ordenadas en orden decreciente a la varianza que representen sobre la salida a predecir, de modo que, limitando las variables a un número menor que las entradas iniciales la pérdida de información relevante sea menor pudiendo aproximar la variable a predecir de un modo más aproximado.

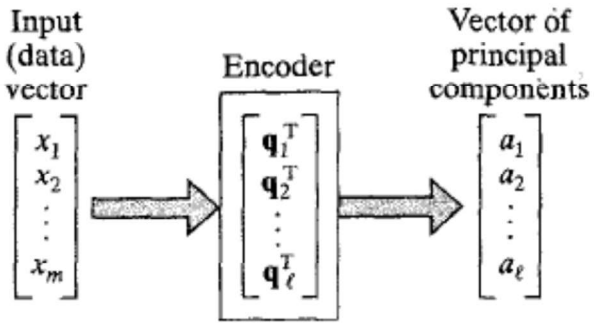

Figura 7.16: Sintetización de componentes principales

Recuperado de (Haykin 1999)

7.4.2.6 Máquinas de vectores de soporte (SVM)

Al igual que las RNA de tipo MLP y las RNA de tipo RBF, las redes de máquinas de vectores de soporte (en lo sucesivo SVM) pueden ser utilizadas para la clasificación de patrones y para aplicaciones de regresión no lineal.

La idea de las SVM consiste en construir un hiperplano o un conjunto de ellos haciendo que con ello se maximice la separación entre los resultados clasificados como dos categorías, y su capacidad reside en, predecir si un nuevo punto pertenece a una categoría o a la otra.

La separación delimitada por un hiperplano es un caso particular para un caso en dos dimensiones pero usualmente se dan otras limitaciones como son:

- Existencia de más de dos variables predictoras.
- Las curvas de separación no tienen por qué ser lineales.
- En ocasiones no pueden separarse totalmente los dos conjuntos de datos.
- Puede darse una clasificación en más de dos categorías.

Como solución a los casos linealmente no separables es necesario utilizar un método que convierta los datos de entrada a un espacio euclidiano de mayor dimensión en el que los datos sean separables por un hiperplano, esto se consigue mediante el uso de las *funciones de kernel.*

7.4.3 Análisis mediante RNA

7.4.3.1 Consideraciones generales para el análisis mediante RNA

Para el análisis mediante RNA la muestra se ha dividido en 3 partes claramente diferenciadas:

1. **Muestra de entrenamiento:** Se tomó esta parte de la muestra como datos con los que entrenar y optimizar el modelo de modo que este adquiera el comportamiento deseado. Se utilizaron los primeros 16400 periodos horarios, es decir, un total de 683 días aproximadamente. Estos datos se han desordenado previamente para un mejor entrenamiento de la red.
2. **Muestra de validación:** Esta parte de la muestra ha sido utilizada para comprobar los resultados que se obtienen para cada una de las iteraciones y, determinar, cuando dejar de entrenar la red. Esta parte de la muestra se compone de los datos entre los periodos 16401 hasta 20824, lo que hace un total de 4423 periodos de 1 hora, lo que hace un total de 184 días aproximadamente.
3. **Muestra de test:** Esta fracción de la muestra se ha utilizado para validar los modelos creados mediante la muestra de entrenamiento y se compone de los datos entre los periodos 20825 y 27851 que hacen un total de 7026 periodos de 1 hora, haciendo un total de 293 días aproximadamente.

- Los modelos se han creado con el programa informático “Neurosolutions”, del cual dispone la Universidad de La Rioja de una licencia para el desarrollo de los modelos. Los indicativos del error de predicción serán: la raíz del error cuadrático medio (*root mean square error* (en lo sucesivo RMSE)) y el error absoluto medio (en lo sucesivo MAE) y el criterio de información de Akaike (en lo sucesivo AIC) cuyas fórmulas son las siguientes:

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |P_i - P_i^*| \tag{7.34}$$

$$RMSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (P_i - P_i^*)^2 \tag{7.35}$$

$$AIC = N \cdot ln(RMSE) + 2 \cdot K \tag{7.36}$$

Siendo:

P_i: *Potencia* real.

P_i^*: Potencia predicha.

N= Número de muestras de test.

K= Número de parámetros, pudiendo ser:

- K= (Nº de neuronas+1)·Nº de entradas + Nº de entradas Para el caso de redes GFFN.
- K= (Nº de neuronas+1)·Nº de entradas Para el resto de las redes.

El análisis de modelos basados en RNA se ha subdividido en dos partes:

1. **Elaboración de modelos sin variables exógenas:** En este apartado únicamente se toman variables relacionadas con la potencia a predecir, como pueden ser: la potencia demandada 24/48/168 horas antes, el día de la semana, etc.
2. **Elaboración de modelos con variables exógenas:** En este apartado, además de las variables tomadas en el apartado anterior, se tomarán variables de tipo meteorológico, que pueden tener cierta relación con la variable a predecir, en este caso, con la potencia demandada.

7.4.3.2 Metodología para la selección óptima de la muestra

Los modelos de RNA anteriormente explicados se utilizarán para predecir el término potencia demandada del sistema a estudiar, pero se deberán tener en cuenta las siguientes premisas:

- Existen multitud de variables explicativas, tanto en el caso del análisis con variables exógenas como en el caso de sin variables exógenas.
- Aparentemente no se conoce cuáles de las variables son más significativas y cuales menos.
- La mayoría de los modelos permiten variar tanto el número de neuronas ocultas de cada capa como la cantidad de capas ocultas de la red.

Un análisis pormenorizado contemplaría la realización de un modelo para cada uno de los diferentes casos, por ejemplo, el modelo creado con todas las variables podría ser diferente al creado con todas las variables menos la variable *a*, y este diferente al creado con todas las variables menos la variable *b* y así hasta todas las variables menos

la variable *n*, y del mismo modo para el caso de todas las variables excepto dos variables incluyendo todas las combinaciones posibles para este caso, y generalizando hasta el caso de una única variable.

Dada la imposibilidad de realización del método anteriormente descrito se plantea lo siguiente:

1. Realizar un modelo de tipo perceptrón multicapa (MLP) con todas las variables de entrada y con un número de neuronas entre 5 y 13.
2. Realizar un análisis de sensibilidad con el fin de evaluar cómo afectan los cambios en las variables en el resultado del modelo final para intentar determinar cuáles son las variables que mayor peso presentan y con ello eliminar las variables menos significativas para intentar simplificar y mejorar el modelo.
3. Comprobación que el modelo anteriormente presentado se comporta igual o mejor que el modelo inicial, pudiéndose dar dos situaciones:
    - El modelo simplificado se comporta igual o mejor que el modelo primitivo: Se realizará un análisis de sensibilidad para determinar, en este caso cuáles son las variables menos significativas, contemplando la eliminación de alguna de ellas a fin de mejorar y simplificar el modelo.
    - El modelo simplificado se comporta peor que el modelo primitivo: Se buscará un modelo intermedio entre el modelo inicial y el modelo simplificado que reduzca el tamaño del modelo inicial y/o lo mejore.
4. Habiendo obtenido un modelo de variables más significativas realizar los análisis para los tipos de RNA estudiados en el *subapartado 7.4.2* con una única capa oculta de neuronas.
5. En caso de que la RNA permita variar el número de neuronas en su capa oculta se realizarán dos análisis, uno con un número de neuronas entre 5 y 13 y otro entre 14 y 20.
6. De cada uno de los dos análisis se evaluará el caso que mejores resultados presente.
7. Los apartados del 1 al 6 se realizarán para los siguientes casos:
    - Análisis sin variables exógenas (en lo sucesivo VE)
    - Análisis con VE.

7.4.3.3 Muestra de datos 1: Con VE

Los resultados del modelo creado con todas las variables disponibles son representados en la *tabla 7.1.*

	Modelo Inicial
AIC	132385.08
RMSE	11891.30
MAE	8353.99

Tabla 7.1: Indicadores de error del modelo realizado con todas las variables

Realizando un análisis de sensibilidad al primer modelo, que utiliza todas las variables disponibles se obtienen los resultados presentados en la *figura 7.17.*

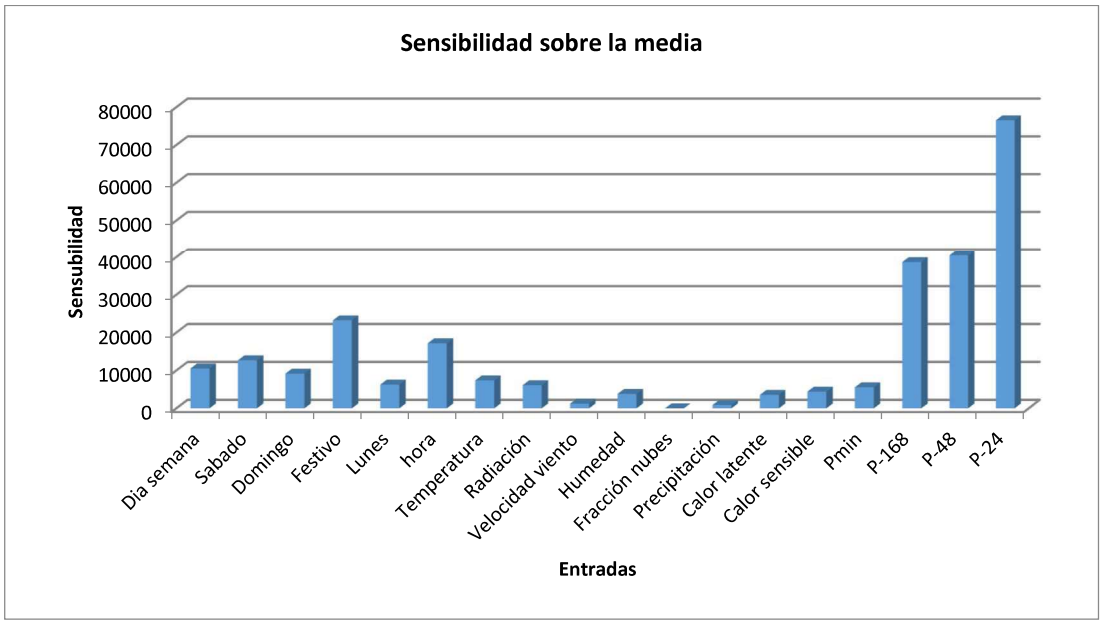

Figura 7.17: Análisis de sensibilidad del modelo realizado con todas las variables

Puede apreciarse que la contribución de determinadas entradas al modelo generado es mínima, por lo que, su evaluación, aparte de complicar el modelo, es posible que lo distorsione. En base a ello se realizarán varios modelos intermedios eliminando las entradas que menor peso tengan en el análisis de sensibilidad de la *figura 7.17* en busca de encontrar un modelo simplificado que se ajuste mejor al término de potencia a predecir.

Tras la realización de varios modelos intermedios se ha decidido eliminar las siguientes variables: Precipitación, fracción de nubes, velocidad de viento y humedad, obteniéndose los resultados presentados en la *tabla 7.2.*

	Modelo inicial	Modelo reducido	Diferencia
AIC	132385.08	132328.77	56.31
RMS	11891.30	11937.78	-46.49
MAE	8353.99	8425.39	-71.41

Tabla 7.2: Comparativa de los indicadores de los 2 modelos estudiados I

Mediante este modelo se obtiene un resultado en términos de error sensiblemente mayor al modelo con todas las variables como puede comprobarse en la *tabla 5.2.*

Pese al leve aumento del error, el valor del término AIC es positivo, lo que significa que, pese a que el término de error es mayor, la calidad del modelo estadístico creado es superior, dado que presenta un número de entradas más reducido que simplifica mucho el modelo a costa de un leve aumento del error.

El análisis de sensibilidad de este modelo es el representado en la *figura 7.1*8. Puede apreciarse que, en este caso se da que las variables con un peso menor están muy equiparadas, por lo que se realizarán varios modelos intermedios eliminando algunas de ellas a fin de determinar con cual se obtienen mejores resultados.

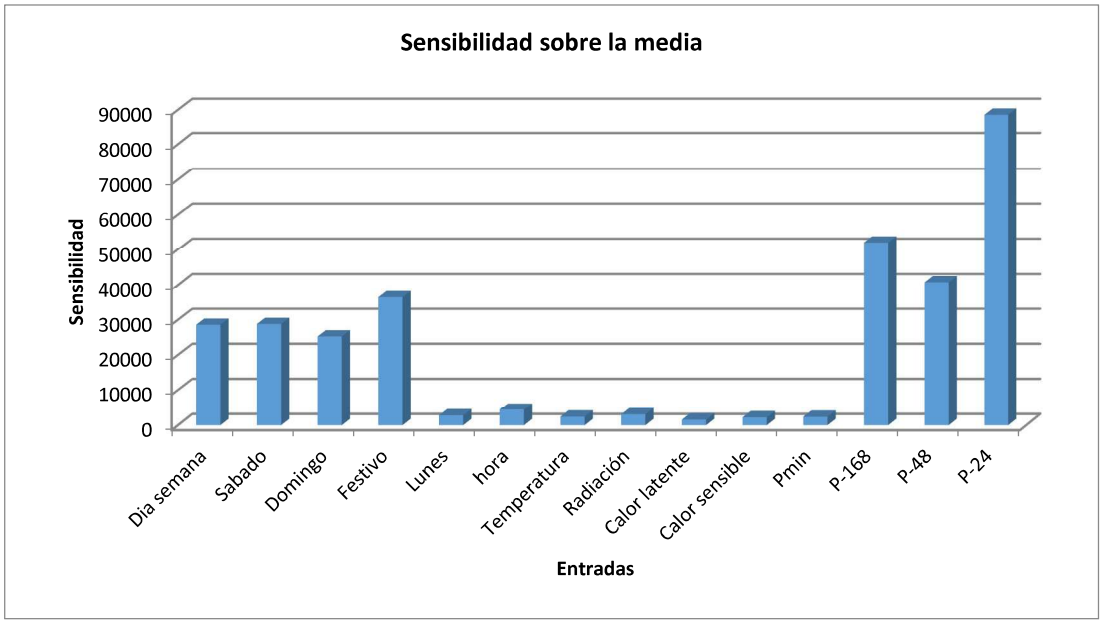

Figura 7.18: Análisis de sensibilidad del modelo reducido

Tomando como referencia los valores de los indicativos del modelo inicial, a fin de establecer una comparativa entre los diferentes modelos estudiados se obtienen los resultados de la *tabla 5.3.*

	Inicial	Reducido	Variante 1	Variante 2	Variante 3	Variante 4
AIC	132385.08	132328.77	132189.99	132333.64	132210.06	132337.49
Diferencia		56.31	195.09	51.44	175.02	47.59
RMSE	11891.30	11937.78	11867.91	12013.33	11908.11	12014.31
Diferencia		-46.49	23.39	-122.03	-16.81	-123.01
MAE	8353.99	8425.39	8341.79	8394.09	8391.95	8412.18
Diferencia		-71.41	12.20	-40.11	-37.96	-58.19

Tabla 7.3: Comparativa de los indicadores de los modelos estudiados II

Conteniendo cada uno de los modelos las siguientes variables:

Variante 1: Día semana, sábado, domingo, lunes, festivo, hora, temperatura, radiación, calor latente, P-168, P-48 y P-24.

Variante 2: Día semana, sábado, domingo, lunes, festivo, hora, temperatura, Pmin, P-168, P-48 y P-24.

Variante 3 Día semana, sábado, domingo, lunes, festivo, hora, temperatura, radiación calor latente, P-168, P-48 y P-24.

Variante 4: Día semana, sábado, domingo, lunes, festivo, hora, temperatura, radiación, Pmin, P-168, P-48 y P-24.

Como puede apreciarse en la *tabla 7.3* el modelo Variante 1 es el que mejor se comporta, presentando un AIC menor en 195,09 unidades al modelo inicial, y, además menores valores de error RMSE y MAE. Por ello, las variables presentes en este modelo serán las evaluadas más adelante con el resto de modelos de RNA.

7.4.3.4 Muestra de datos 2: Sin VE

Teniendo en cuenta únicamente las siguientes variables: Día de la semana, sábado, domingo, festivo, lunes, hora, potencia mínima, potencia 168h antes, potencia 48h antes y potencia 24h antes se ha realizado un modelo el cual presenta los resultados presentes en la *tabla 7.4.*

	Muestra inicial
RMS	11796.00
MAE	8183.00
AIC	132079.73

Tabla 7.4: Indicadores del modelo realizado con todas las variables

El análisis de sensibilidad sobre la media del modelo creado es el representado en la *figura 7.19.*

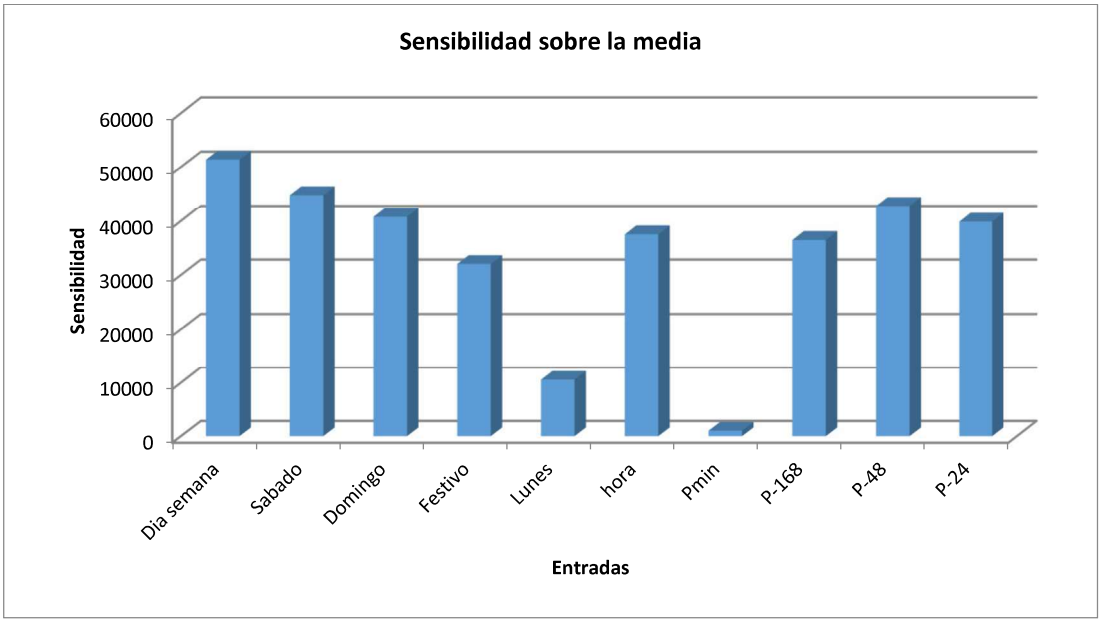

Figura 7.19: Análisis de sensibilidad del modelo con todas las variables

En base al análisis de sensibilidad realizado se ha realizado un modelo adicional sin tener en cuenta el término de potencia mínima para comprobar si el modelo mejora los resultados del anterior. La comparativa propuesta es analizada en la *tabla 7.5.*

	Inicial	Sin Pmin	Diferencia
AIC	132079.73	132660.29	-580.56
RMS	11796.00	12114.98	-318.98
MAE	8183.00	8348.95	-165.95

Tabla 7.5: Comparativa entre los dos modelos realizados

Como puede apreciarse, los resultados empeoran notablemente el modelo anteriormente propuesto, por lo que, se utilizará como modelo estándar para el análisis sin VE el modelo que incluye la variable potencia mínima.

7.5 Modelos elaborados mediante Matlab

El programa informático Matlab ha sido utilizado para la elaboración de modelos basados en lógica difusa, por lo que, en los sucesivos apartados se introducirá el concepto de la lógica difusa, las consideraciones utilizadas para la subdivisión de la muestra de datos y el código utilizado.

7.5.1 Marco teórico

La lógica difusa se centra en el trabajo con conjuntos de datos cuyos límites no están perfectamente definidos, es decir, la función que determina si un parámetro pertenece o no a un conjunto es una función continua y gradual, ejemplo *figura 7.20*. Los sistemas basados en lógica difusa intentan imitar la forma en la que razonan los humanos, pero de una forma mucho más rápida y tienen la capacidad de ser intolerantes a ruidos en los datos de entrada.

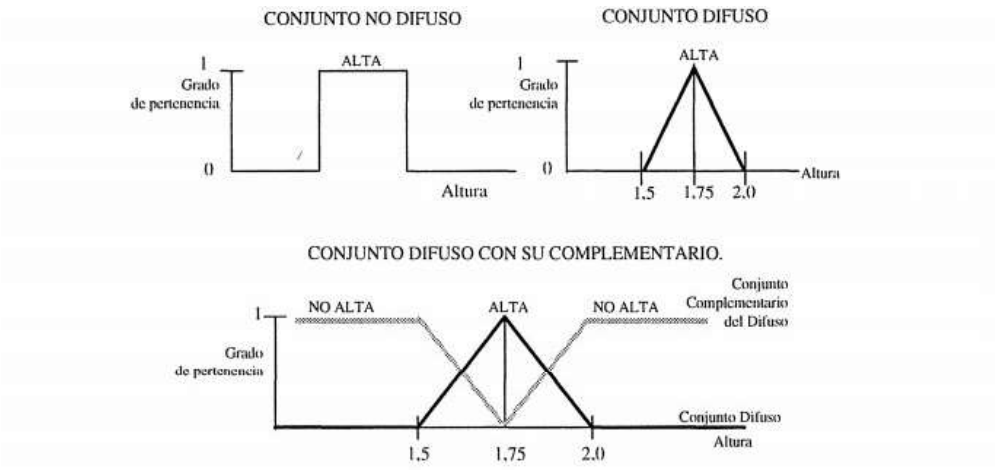

Figura 7.20: Representación del grado de pertenencia en conjuntos difusos y no difusos. Recuperado de (A.Pazos & C.M.Dans 1996)

Los sistemas de lógica difusa aplicada a RNA potencian a estas últimas en los siguientes aspectos:

- Mejora la capacidad de representación de las RNA, dado que, mediante la lógica difusa se pueden contemplar valores no definidos, o con cierto grado de imprecisión.
- Se precisa un menor número de datos para el entrenamiento, disminuyendo el tiempo y requerimientos de las máquinas para la elaboración de modelos.

7.5.2 Análisis mediante lógica difusa

7.5.2.1 Consideraciones para el análisis mediante lógica difusa

Para el análisis mediante lógica difusa la muestra se dividirá en tres partes (entrenamiento, validación y test) conforme se ha explicado en el subapartado 7.4.3.1 y para la evaluación de resultados se utilizarán los indicadores del error citados en ese mismo subapartado (MAE y RMSE). Del mismo modo que en análisis anteriores se trabajará con dos muestras de datos, una que incluye VE y otra que no lo hace.

El método utilizado para la selección óptima de las variables de la muestra ha sido el explicado en el subapartado 7.4.3.2 y las variables utilizadas para este análisis han sido las resultantes de los análisis de sensibilidad realizados en los subapartados 7.4.3.3 y 7.4.3.4.

El análisis mediante lógica difusa se realizará mediante Matlab con el código explicado en el siguiente subapartado.

7.5.3 Código utilizado para el análisis mediante lógica difusa con Matlab

7.5.3.1 Modelo sin VE

```
%En primer lugar se construirá una matriz de 11 columnas y 27850 filas)
SIN_VE=[Diasemana,Sabado,Domingo,Festivo,Lunes,hora,Pmin,P168,P48,P24,Suma
P];

%Se divide la matriz en 3 sub-matrices, una de entrenamiento, otra de validación de
este entrenamiento y otra de test
train=SIN_VE(1:16400,1:11);
cross=SIN_VE(16401:20824,1:11);
test=SIN_VE(20825:27850,1:11);

%Se separan las variables de entrada y la variable salida para cada una de las muestras
de trabajo
variables_train= SIN_VE(1:16400,1:10);
sal_train= SIN_VE(1:16400,11);
variables_cross= SIN_VE(16401:20824,1:10);
sal_cross= SIN_VE(16401:20824,11);
variables_test= SIN_VE(20825:27850,1:10);
sal_test= SIN_VE(20825:27850,11);


%Se generan agrupaciones a partir de los datos de entrenamiento fijando como
parámetro el radio de estas. El radio se utiliza para crear agrupaciones de resultados a
los que se aplicará un tratamiento similar
```

```
fis_1 = genfis2(variables_train,sal_train,1);

%Determinación de las características del entrenamiento
trnOpt = [100 0 0.01 0.9 1.1];
dispOpt = [1 1 1 1];
%siendo:
%TrnOpt (1): Número de iteraciones de entrenamiento.
%TrnOpt (2): Meta del error de entrenamiento.
%TrnOpt (3): Valor del peso inicial.
%TrnOpt (4): Valor de decremento en cada iteración.
%TrnOpt (5): Valor de incremento en cada iteración.
%Ejecución de la función anfis para el entrenamiento de la red utilizando como muestra de entrenamiento los datos comprendidos entre 1-16400 y como validación 16001-20484
[fis,error,stepsize,chkFis,chkErr] = anfis(train, fis_1,trnOpt,dispOpt,cross);

%Evaluación del modelo creado fis_1 mediante la función evalfis de los resultados en cada una de las muestras de trabajo
resultado_train= evalfis(variables_train, fis_1);
resultado_cross= evalfis(variables_cross, fis_1);
resultado_test = evalfis(variables_test, fis_1);

%Cálculo del error RMSE de los resultados obtenidos el ejecutar el modelo fis_1 respecto a los valores reales de demanda
comparativa_train=[resultado_train,sal_train,(resultado_train-sal_train).^2];
comparativa_cross=[resultado_cross,sal_cross,(resultado_cross-sal_cross).^2];
comparativa_test=[resultado_test,sal_test,(resultado_test-sal_test).^2];
Suma_train=sum(comparativa_train)
RMSE_train=sqrt(Suma_train(1,3)/16400)
Suma_cross=sum(comparativa_cross)
RMSE_cross=sqrt(Suma_cross(1,3)/4424)
Suma_test=sum(comparativa_test)
RMSE_test=sqrt(Suma_test(1,3)/7026)

%Cálculo del error MAE de los resultados obtenidos el ejecutar el modelo fis_1 respecto a los valores reales de demanda
```

```
comparativa_train2=[resultado_train,sal_train,abs(resultado_train-sal_train)];

comparativa_cross2=[resultado_cross,sal_cross,abs(resultado_cross-sal_cross)];

comparativa_test2=[resultado_test,sal_test,abs(resultado_test-sal_test)];

Suma_train2=sum(comparativa_train2)

MAE_train=Suma_train2(1,3)/16400

Suma_cross2=sum(comparativa_cross2)

MAE_cross=Suma_cross2(1,3)/4424

Suma_test2=sum(comparativa_test2)

MAE_test=Suma_test2(1,3)/7026


%Resultados

MAE=[MAE_train,MAE_cross,MAE_test]

RMSE=[RMSE_train,RMSE_cross,RMSE_test]

Resultados=[RMSE,MAE]
```

7.5.3.2 Modelo con VE

```
%En primer lugar se construirá una matriz de 13 columnas y 27850 filas)
CON_VE=[Diasemana,Sabado,Domingo,Festivo,Lunes,hora,Calorsensible,Radiacin,T
emperatura,P168,P48,P24,SumaP];

%Se divide la matriz en 3 sub-matrices, una de entrenamiento, otra de validación de
este entrenamiento y otra de test
train=CON_VE(1:16400,1:13);
cross=CON_VE(16401:20824,1:13);
test=CON_VE(20825:27850,1:13);

%Se separan las variables de entrada y la variable salida para cada una de las muestras
de trabajo
variables_train= CON_VE(1:16400,1:12);
sal_train= CON_VE(1:16400,13);
variables_cross= CON_VE(16401:20824,1:12);
sal_cross= CON_VE(16401:20824,13);
variables_test= CON_VE(20825:27850,1:12);
sal_test= CON_VE(20825:27850,13);


%Se generan agrupaciones a partir de los datos de entrenamiento fijando como
parámetro el radio de estas. El radio se utiliza para crear agrupaciones de resultados a
los que se aplicará un tratamiento similar
fis_1 = genfis2(variables_train,sal_train,1);
```

```
%Determinación de las características del entrenamiento
trnOpt = [100 0 0.01 0.9 1.1];
dispOpt = [1 1 1 1];
%siendo:
%TrnOpt (1): Número de iteraciones de entrenamiento.
%TrnOpt (2): Meta del error de entrenamiento.
%TrnOpt (3): Valor del peso inicial.
%TrnOpt (4): Valor de decremento en cada iteración.
%TrnOpt (5): Valor de incremento en cada iteración.

%Ejecución de la función anfis para el entrenamiento de la red utilizando como muestra de entrenamiento los datos comprendidos entre 1-16400 y como validación 16001-20484
[fis,error,stepsize,chkFis,chkErr] = anfis(train, fis_1,trnOpt,dispOpt,cross);

%Evaluación del modelo creado fis_1 mediante la función evalfis de los resultados en cada una de las muestras de trabajo
resultado_train= evalfis(variables_train, fis_1);
resultado_cross= evalfis(variables_cross, fis_1);
resultado_test = evalfis(variables_test, fis_1);

%Cálculo del error RMSE de los resultados obtenidos el ejecutar el modelo fis_1 respecto a los valores reales de demanda
comparativa_train=[resultado_train,sal_train,(resultado_train-sal_train).^2];
comparativa_cross=[resultado_cross,sal_cross,(resultado_cross-sal_cross).^2];
comparativa_test=[resultado_test,sal_test,(resultado_test-sal_test).^2];
Suma_train=sum(comparativa_train)
RMSE_train=sqrt(Suma_train(1,3)/16400)
Suma_cross=sum(comparativa_cross)
RMSE_cross=sqrt(Suma_cross(1,3)/4424)
Suma_test=sum(comparativa_test)
RMSE_test=sqrt(Suma_test(1,3)/7026)

%Cálculo del error MAE de los resultados obtenidos el ejecutar el modelo fis_1 respecto a los valores reales de demanda
```

```
comparativa_train2=[resultado_train,sal_train,abs(resultado_train-sal_train)];
comparativa_cross2=[resultado_cross,sal_cross,abs(resultado_cross-sal_cross)];
comparativa_test2=[resultado_test,sal_test,abs(resultado_test-sal_test)];
Suma_train2=sum(comparativa_train2)
MAE_train=Suma_train2(1,3)/16400
Suma_cross2=sum(comparativa_cross2)
MAE_cross=Suma_cross2(1,3)/4424
Suma_test2=sum(comparativa_test2)
MAE_test=Suma_test2(1,3)/7026

%Resultados
MAE=[MAE_train,MAE_cross,MAE_test]
RMSE=[RMSE_train,RMSE_cross,RMSE_test]
Resultados=[RMSE,MAE]
```

7.6 Modelos realizados mediante RStudio

Los modelos de predicción creados mediante Rstudio son de los siguientes tipos:

- Regresión lineal.
- Redes neuronales.
- Bosques aleatorios (Random Forests) basados en árboles de decisión.

Dado que en apartados anteriores ya se ha hablado sobre regresión lineal y redes neuronales, en los siguientes apartados se explicará en que se basan los árboles de decisión y los bosques aleatorios.

7.6.1 Marco teórico

7.6.1.1 Arboles de decisión

Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de manera que la decisión final se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas.

Los árboles en los que la variable destino está formada por un conjunto finito de valores se denominan árboles de clasificación y los árboles en los que la variable destino puede tomar valores continuos se llaman árboles de regresión.

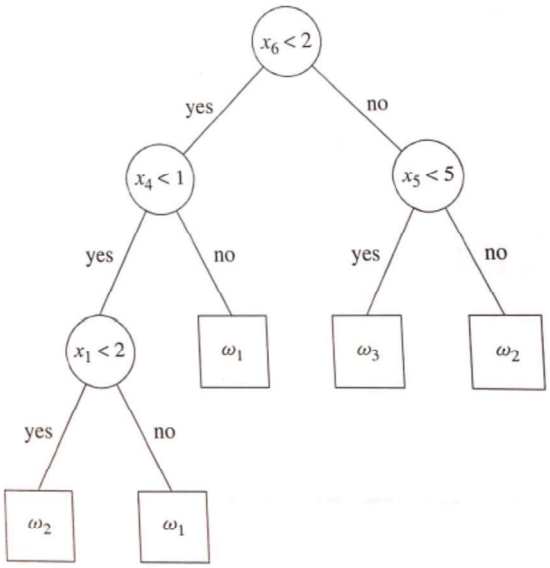

Figura 7.21: Árbol de decisión.
Recuperado de (Webb. A. (2002))

Los árboles de decisión están compuestos de:

Nodo raíz: Solo existirá un nodo raíz por cada árbol predictor y este nodo no posee nodos de entrada.

Nodos interiores: Contienen preguntas sobre atributos concretos.

Ramas: Cada rama constituye una respuesta/valor del atributo.

Nodos hoja: Cada nodo hoja se refiere a una predicción. En función del tipo de árbol predictor podrán ser de dos tipos:

- Clasificación: Etiqueta de una clase.
- Regresión: Función de valores continuos.

7.6.1.2 Poda en arboles de decisión

Los árboles de decisión son capaces de crear un modelo completo y consistente con respecto a los datos de entrenamiento. En primer lugar, esto hace que éste se ajuste demasiado a los ejemplos ya vistos sobreentrenando el modelo y haciendo que este sea demasiado específico y poco general, haciendo que los resultados obtenidos con nuevas muestras difieran mucho de los resultados obtenidos con los datos de entrenamiento. En segundo lugar, cuando el vector de entrenamiento contiene ruido (errores en los atributos), el modelo intenta ajustarse a esos errores generando un mal comportamiento y perjudicando el modelo.

La forma más usual de evitar el sobreentrenamiento es modificar los algoritmos de modo que estos se comporten de un modo más general, lo que se consigue eliminando condiciones de las ramas del árbol, en el caso de los árboles de decisión este procedimiento es llamado “poda”.

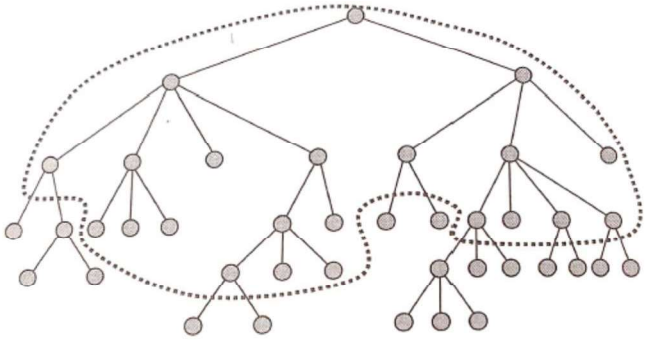

Figura 7.22: Ejemplo de poda para la eliminación de nodos inferiores. Recuperado de(Hernández-Orallo, J., Ramírez, M.J., Ferri, C(2004))

7.6.1.3 Bagging y boosting

El bagging y el boosting se centran en usar varios modelos individuales para desarrollar un mejor modelo en base a los anteriores. El bagging da, a todos ellos, un peso igual. Por el contrario, en el boosting, se da diferentes pesos a los modelos en función de su predominio.

Se puede pensar que, creando varios árboles de una misma muestra, estos serán prácticamente idénticos, cosa que no es así, dado que los conjuntos de entrenamiento utilizados para cada árbol pueden tener atributos diferentes.

Estos métodos tienen como propósito la reducción de la varianza, mediante la creación de muchos modelos individuales antes de crear un modelo definitivo.

7.6.1.4 Bosques aleatorios (Random Forests)

Los bosques aleatorios (Random Forests) consisten en construir multitud de árboles de decisión distintos y, a partir de la combinación de ellos obtener un árbol nuevo, es decir, una modificación del *Bagging*. El error de generalización converge hacia un límite conforme la muestra de árboles en el bosque aumenta.

Cada árbol depende de un vector aleatorio de la muestra y los siguientes árboles se generan mediante un remuestreo con reposición (bootstrap), esta selección aleatoria hace que el árbol generado sea más robusto con respecto al ruido, incluso cuando algunas de las variables no son representativas.

7.6.2 Consideraciones para el análisis mediante RStudio

Para el análisis mediante RStudio la muestra se ha dividido en 2 partes claramente diferenciadas:

1. **Muestra de entrenamiento:** Se tomó esta parte de la muestra como datos con los que entrenar y optimizar el modelo de modo que este adquiera el comportamiento deseado. Se utilizaron los primeros 20825 periodos horarios, es decir, un total de 868 días aproximadamente. Estos datos se han desordenado previamente para un mejor entrenamiento de la red.
2. **Muestra de test:** Esta fracción de la muestra se ha utilizado para validar los modelos creados mediante la muestra de entrenamiento y se compone de los datos entre los periodos 20826 y 27851 que hacen un total de 7026 periodos de 1 hora, haciendo un total de 293 días aproximadamente.

Internamente, en la muestra de entrenamiento se realizará una subdivisión en muestra de entrenamiento pura y muestra de validación cruzada del siguiente modo:

- Se subdividirá la muestra de entrenamiento en 5 partes.
- Aleatoriamente, 4 de esas partes serán utilizadas como muestra de entrenamiento y la restante como muestra de validación cruzada. Este proceso se realizará un total de 3 veces.

El método utilizado para la selección óptima de las variables de la muestra ha sido el explicado en el subapartado 7.4.3.2 y las variables utilizadas para este análisis han sido las resultantes de los análisis de sensibilidad realizados en los subapartados 7.4.3.3 y 7.4.3.4.

Los indicativos del error de predicción serán: RMSE, MAE y el coeficiente de correlación de Pearson cuyas fórmulas son las siguientes:

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |P_i - P_i^*| \qquad (7.37)$$

$$\boldsymbol{RMSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (P_i - P_i^*)^2} \qquad (7.38)$$

$$R^2 = \frac{{\sigma^2}_{XY}}{{\sigma^2}_X \cdot {\sigma^2}_Y} \quad (7.39)$$

siendo:

σ_{XY}: Covarianza de X,Y.

σ_X: *Desviación estándar de la variable X.*

σ_Y: *Desviación estándar de la variable Y.*

El análisis mediante lógica difusa se realizará mediante RStudio con el código explicado en el siguiente subapartado.

7.6.3 Código utilizado para el análisis mediante RStudio

Para realizar el estudio tanto con la muestra con VE como con la muestra sin VE únicamente se deberá editar la ruta del archivo del cual se toman los valores.

Para realizar el análisis con los diferentes modelos bastará con introducir el nombre del modelo y las librerías necesarias para cada uno de ellos.

```
# Introducir las librerías necesarias según el modelo
library(ggplot2)
library(ranger)
library(readxl)
library(doParallel)
library(randomForest)
library(xlsx)
library(Cubist)
library(ranger)
library(Rborist)
library(xgboost)
library(caret)
…
rsq <- function (x, y) cor(x, y) ^ 2
#Ruta del archivo, con editar únicamente la ruta se puede realizar el análisis para los modelos con VE y sin VE
setwd("F:/demanda1/4 Modelos R/Con_VE")
```

```
data <- read_excel("CON_VE.xlsx", sheet = "Hoja1")
data<-data[1:27850,c(2:ncol(data))]
training <- data[1:20825,c(1:ncol(data))]
testing <- data[20826:27850,c(1:ncol(data))]
# Función para normalizar entre [0-1]
normaliza.minmax <- function(x) {(x-min(x))/(max(x)-min(x))}
#Normalización de todas las variables de entrada
data.new <- as.data.frame(data)
data.new[,1:(ncol(data.new)-1)] <- apply(data.new[,1:(ncol(data.new)-
1)],2,normaliza.minmax)


cl <- makePSOCKcluster(7)
set.seed(12)
registerDoParallel(cl)
#División de los datos de entrenamiento en 5 particiones, usadas 4 como
entrenamiento y 1 como validación cruzada. Se repetirá este proceso 3 veces
fitControl <- trainControl(## 5-fold CV
  method = "repeatedcv",
  number = 5,
  ## repetido 3 veces
  repeats = 3)
#Especificación del modelo a utilizar
s <- "lm"
#Asignación de los datos normalizados a los vectores de entrenamiento y test, en el
caso de no querer utilizar datos normalizados escribir # en las dos líneas siguientes
training<-data.new[1:20825,]
testing<-data.new[20826:27850,]
#Entrenamiento del modelo, para cambiar algún parámetro respecto a los estándares
del modelo escribirlo dentro del bucle.
model <- train(Suma ~ ., data = training,
               method = s,
               preProcess = c("center", "scale"),
               trControl = fitControl)


stopCluster(cl)
model
```

```
sink(paste0("Modelo_",s,".txt"))
print(model)
print("\n")
print("\n")
sal_train <- predict(model, training)
error <- sal_train-training$Suma
rmse_training <- sqrt(mean((error)^2))
print(paste0("RMSE entrenamiento: ", rmse_training))
mae_training <- mean(abs(error))
print(paste0("MAE entrenamiento: ", mae_training))
rsq_training <- rsq(sal_train,training$suma)
print(paste0("R2 entrenamiento: ", rsq_training))
sal_test <- predict(model, testing)
error <- sal_test-testing$Suma
rmse_testing <- sqrt(mean((error)^2))
print(paste0("RMSE test: ", rmse_testing))
mae_testing <- mean(abs(error))
print(paste0("MAE testing: ", mae_testing))
rsq_testing <- rsq(sal_test,testing$Suma)
print(paste0("R2 testing: ", rsq_testing))
print("\n")
print("\n")
print(summary(model))
sink()

file_e <- paste0("Salida_", s, ".xlsx")
# Recompone los ceros para tener todas las horas. Primero dimensiono vectores
salida_1 <- matrix(training$Suma,nrow=20825,ncol=2)
salida_2 <- matrix(testing$Suma,nrow=7025,ncol=2)
for (i in c(1:20825)) {
  salida_1[i,2]=sal_train[i]
}
for (i in c(1:7025)) {
  salida_2[i,2]=sal_test[i]
```

```
}
# Calcula algunas métricas adicionales del error("CORR", "MAE", "RMSE", "RAE", "RRSE", "MEE", "SDE", "RSDE")
modelErrors <- function(predicted, actual) {
  sal <- vector(mode="numeric", length=8)
  names(sal) <- c("CORR", "MAE", "RMSE", "RAE", "RRSE", "MEE", "SDE", "RSDE")

  meanPredicted <- mean(predicted,na.rm=T)
  meanActual <- mean(actual,na.rm=T)
  sumPred <- sum((predicted - meanPredicted)^2)
  sumActual <- sum((actual - meanActual)^2)
  sumPredAct <- crossprod(predicted - meanPredicted, actual - meanActual)
  n<- length(actual)
  sal[1] <- sumPredAct / sqrt(sumPred*sumActual)
  sal[2] <- mean(abs(predicted - actual))
  sal[3] <- sqrt(sum((predicted - actual)^2)/n)
  sal[4] <- sum(abs(predicted-actual))/sum(abs(actual-meanActual))
  sal[5] <- sqrt(sum((predicted-actual)^2)/sum((actual-meanActual)^2))
  sal[6] <- mean(predicted - actual)
  sal[7] <- sd(predicted - actual)
  sal[8] <- sd((predicted - actual)/actual,na.rm=T)
  sal
}
# Cálculo de los errores
modelErrors(predicted = salida_1[,2], actual = salida_1[,1])
modelErrors(predicted = salida_2[,2], actual = salida_2[,1])


# Representación gráfica de los datos de predicción frente a los reales de los datos de entrenamiento y test (solo se mostrará el último gráfico ejecutado)
plot(x=salida_1[,1], y=salida_1[,2], xlab="Valor real", ylab="Valor predicho",main="Representación de valores reales frente a predichos")
plot(x=salida_2[,1], y=salida_2[,2], xlab="Valor real", ylab="Valor predicho",main="Representación de valores reales frente a predichos")
# Una predicción perfecta mostraría los puntos sobre la línea roja
lines(x=range(c(salida_1[,1], y=salida_1[,2])),y=range(c(salida_1[,1], y=salida_1[,2])),col="red")
```

```
#Exportación de resultados a Excel
write.xlsx(salida_1, file=file_e, sheetName = "Entrenamiento",
        col.names = TRUE, row.names = FALSE, append = FALSE)
write.xlsx(salida_2, file=file_e, sheetName = "Test",
        col.names = TRUE, row.names = FALSE, append = TRUE)
```

7.6.4 Aplicación de la poda en los árboles de regresión

En base a que los modelos de bosques aleatorios tienden a sobreentrenar, se plantea realizar una poda en los árboles de regresión de los modelos.

Se cambiará únicamente un parámetro por ejecución del algoritmo, de este modo se obtendrán tantos resultados como análisis se realicen.

Como ventaja se tiene que se obtienen los resultados intermedios de todas las iteraciones realizadas, por lo que, puede apreciarse de un mejor modo la variación de los resultados en el grupo de test en función de la variación de los parámetros, sin embargo, es un método lento y costoso.

Los modelos elegidos para el primer experimento son los siguientes: “ranger” y “rf”.

Tras analizar los argumentos modificables en estos modelos se valorarán los siguientes:

- Num.trees*: Número de árboles de regresión del bosque aleatorio. Por defecto 500.
- Min.node.size*: Tamaño mínimo del nodo final (hojas del árbol). Por defecto 5 para regresión.
- Mtry*: Número de variables a dividir cada nodo. Por defecto es la raíz cuadrada (redondeada hacia debajo) de las variables numéricas.

* Denominación para el modelo “ranger”.

7.6.4.1 Metodología

En el análisis del modelo “rf” se variará el tamaño mínimo de nodo desde 10 hasta 40 y la variable que controla el número de variables a dividir cada nodo se elegirá automáticamente por el algoritmo.

En el análisis del modelo "ranger" se variará el tamaño mínimo de nodo desde 5 hasta 30 y la variable que controla el número de variables a dividir cada nodo se elegirá automáticamente por el algoritmo.

7.6.5 Optimización de los modelos de bosques aleatorios

Teniendo en cuenta que en los experimentos realizados aplicando los métodos de poda no mejoran los resultados obtenidos utilizando los parámetros por defecto de los modelos se plantea intentar optimizar estos variando varios de estos parámetros simultáneamente.

El modelo elegido es el modelo "ranger", ya que es el que mejores resultados aporta además de que es uno de los más rápidos en cuanto a su ejecución.

Se realizará un mallado mediante el cual, con una única iteración se probarán multitud de parámetros y el algoritmo elegirá por sí mismo el que mejores resultados proporcione.

La ventaja de este método es que con él se puede realizar un mallado con un número de parámetros grande pero la elección del mejor modelo es realizada por el algoritmo en sí, y este califica como mejor modelo el que tenga un menor valor de RMSE en el grupo de entrenamiento.

El código utilizado será el explicado en el apartado 7.6.3 añadiendo lo siguiente:

```
Grid <-  expand.grid(mtry = 3:12, # modifica el parámetro mtry desde 3 hasta 12
                     splitrule = "variance",
                     min.node.size = (1:8)*5) # modifica el parámetro min.node.size desde 5 hasta 40 con saltos de 5 unidades
model <- train(Suma ~ ., data = training,
               method = s,
                preProcess = c("center", "scale"),
               num.trees=500,
               tuneGrid = Grid, # hace que se aplique el mallado anterior
                trControl = fitControl)
```

Con el código anterior se plantea un mallado que abarca 80 modelos diferentes combinando los valores de las variables "mtry" y "min.node.size".

8. Resultados

8.1 Modelos creados con Excel

Los resultados en términos de medio absoluto (MAE) error porcentual absoluto medio (MAPE) y error cuadrático medio (RMS) para las muestras de entrenamiento y test son los siguientes:

Modelos/Error	Entrenamiento			Test		
	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE
Día anterior	19685.38	13.739%	33638.50	21418.96	14.144%	36411.76
Día similar	13668.95	9.517%	23257.40	14184.53	9.452%	24336.45
Regresivo V2	19554.52	13.552%	33441.26	20836.47	13.690%	35736.66
Regresivo V2	19447.11	13.528%	33103.53	20745.52	13.655%	35510.73
Regresivo V3	12622.89	8.735%	20931.14	13299.05	8.798%	22162.68
Regresivo V4	12639.70	8.769%	21090.49	13470.37	8.857%	22317.06
Regresivo V5	12639.72	8.723%	20968.95	13338.37	8.779%	22208.46
Regresivo V6	12638.26	8.722%	20964.49	13336.50	8.779%	22205.87

Tabla 8.1: Errores en los modelos lineales

Dado que los resultados obtenidos en las muestras de entrenamiento y evaluación son prácticamente iguales, únicamente se presentarán representados los errores MAPE y RMSE de la muestra de evaluación en las *figuras 8.1 y 8.2*.

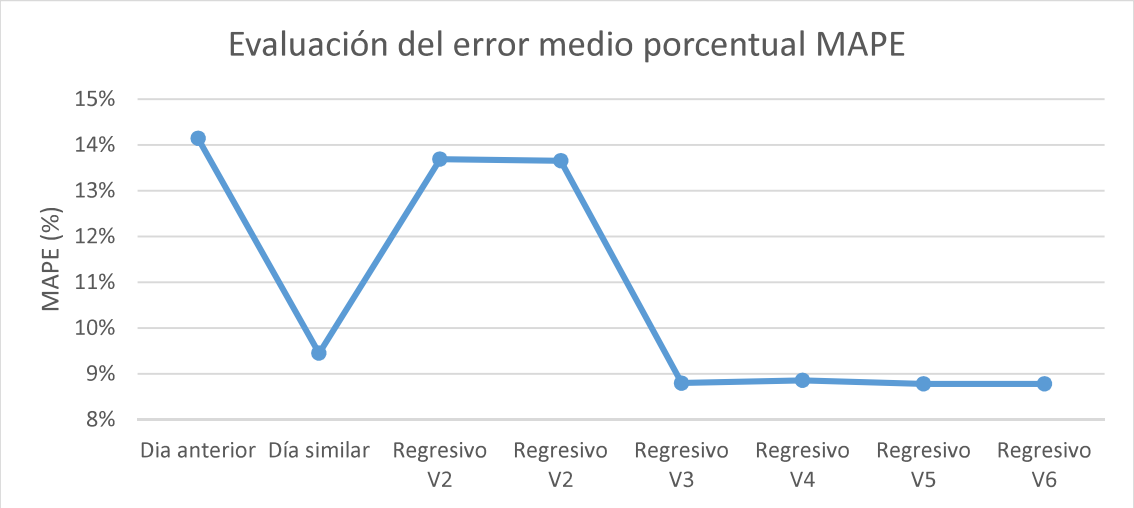

Figura 8.1: Representación del error MAPE (%) en los modelos lineales

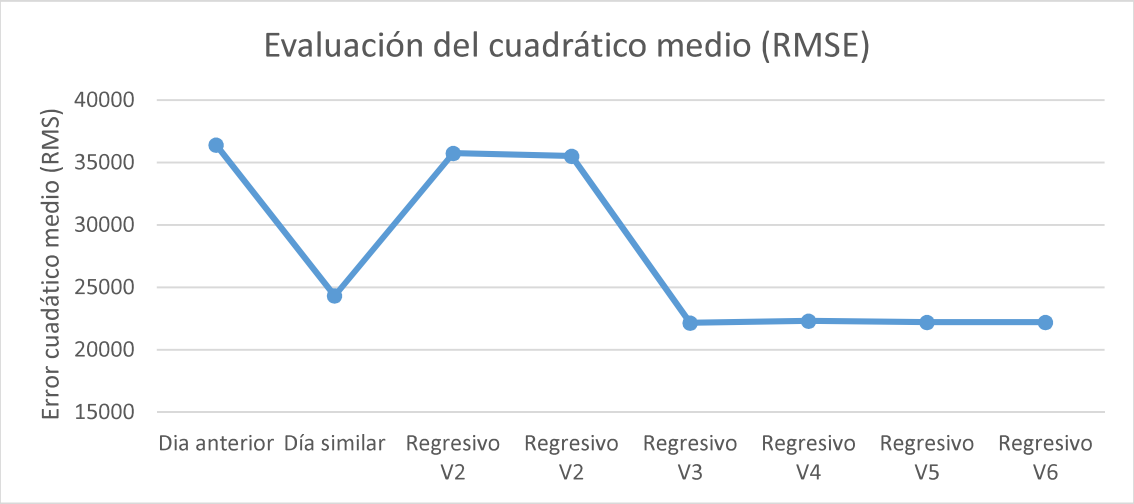

Figura 8.2: Representación del error RMSE en los modelos lineales

8.2 Modelos creados con Neurosolutions

8.2.1 Sin VE

Dado que los análisis de modelos se han realizado en dos grupos, de 5 a 13 y de 14 a 20 neuronas en la primera capa oculta, solo se presentará, en cada caso, el modelo que mejores resultados ofrezca.

Tipo de red neuronal	MLP		Diferencia	GFFN		Diferencia
Rango de neuronas	5-13	14-20		5-13	14-20	
Nº de neuronas	13	20		10	19	
RMSE	11796,61	11931,41	-134,80	12331,86	12028,73	303,13
MAE	8183,08	8276,69	-93,60	8455,67	8308,96	146,71
AIC	132051,73	132351,35	-299,62	132637,61	132466,25	171,36
R^2	0,9675	0,9673	0,0002	0,9653	0,9663	-0,0010
Tipo de red neuronal	PCA		Diferencia	RBF		Diferencia
Rango de neuronas	5-13	14-20		5-13	14-20	
Nº de neuronas	11	17		12	14	
RMSE	13187,16	13151,69	35,47	13766,66	13635,27	131,38
MAE	8827,16	8788,03	39,12	9356,57	9397,28	-40,71
AIC	133579,35	133661,62	-82,27	134203,98	134110,82	93,16
R^2	0,9595	0,9597	-0,0002	0,9558	0,8772	0,0786

Tabla 8.2: Indicativos de los diferentes modelos de RNA sin VE

Los modelos SVM probados presentaban unos resultados notablemente peores que el resto de redes neuronales estudiadas por lo que estos no se han incluido en los resultados.

Sorprendentemente el mejor modelo de los presentados es el modelo MLP con 13 neuronas en la primera capa oculta, siendo este mejor que los obtenidos con un rango de 14 a 20 neuronas.

Las columnas "diferencia" presentan la resta entre los términos RMSE, MAE y AIC para el estudio con 5-13 neuronas frente al estudio con 14-20 neuronas en la primera capa oculta, pudiéndose apreciar que, excepto en el modelo MLP los modelos con un número de neuronas mayor en la primera capa oculta presentan mejores resultados que con un número de neuronas menor.

En las *figuras 8.3 y 8.4* se pueden observar dos representaciones diferentes de las potencias reales frente a las predichas mediante una red neuronal de tipo MLP de 1 capa oculta con 20 neuronas en esa primera capa.

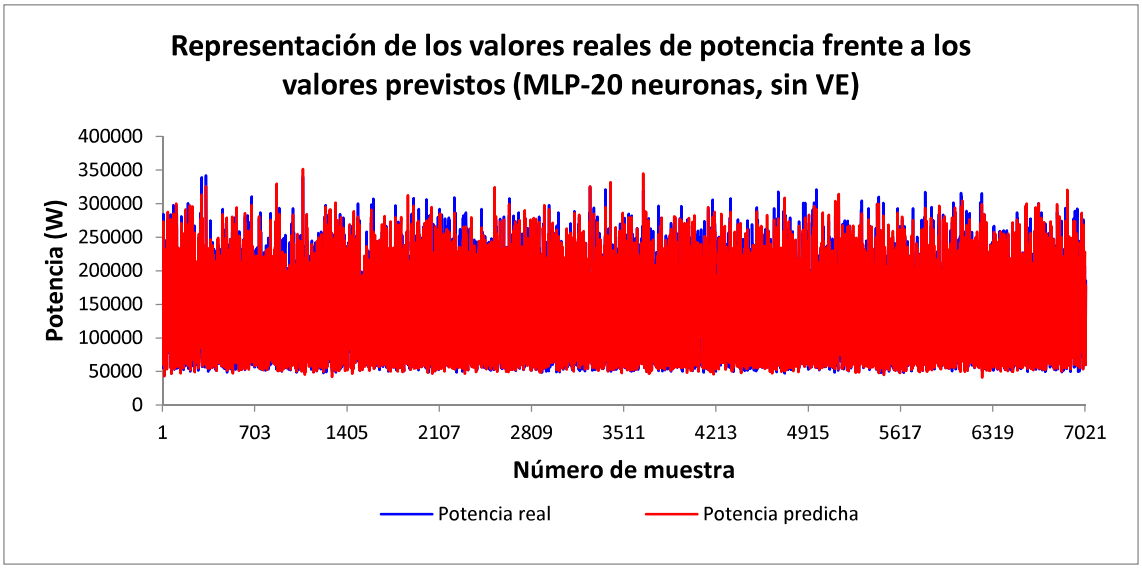

Figura 8.3: Representación de los valores reales de potencia frente a los valores previstos en con un modelo MLP con 20 neuronas en su primera capa oculta (Sin VE) I

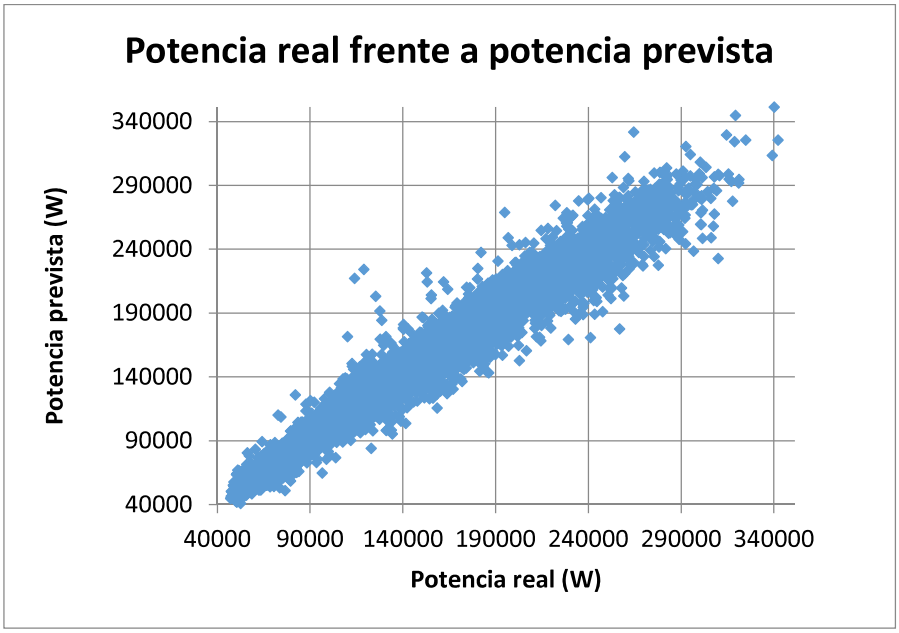

Figura 8.4: Representación de los valores reales de potencia frente a los valores previstos en con un modelo MLP con 20 neuronas en su primera capa oculta (Sin VE) II

8.2.2 Con VE

Dado que los análisis de modelos se han realizado en dos grupos, de 5 a 13 y de 14 a 20 neuronas en la primera capa oculta, solo se presentará, en cada caso, el modelo que mejores resultados ofrezca.

Tipo de red neuronal	MLP		Diferencia	GFFN		Diferencia
Rango de neuronas	5-13	14-20		5-13	14-20	
Nº de neuronas	13	19		12	14	
RMSE	11867,91	11777,05	90,86	12083,86	11859,99	223,87
MAE	8341,79	8249,51	92,28	8514,07	8338,35	175,72
AIC	132189,99	132082,71	107,28	132441,91	132229,05	212,86
R^2	0,9671	0,9677	-0,0005	0,9660	0,9677	-0,0017
Tipo de red neuronal	PCA		Diferencia	RBF		Diferencia
Rango de neuronas	5-13	14-20		5-13	14-20	
Nº de neuronas	13	15		11	16	
RMSE	14086,57	13635,08	451,49	14474,37	13515,23	959,14
MAE	9830,66	9472,85	357,81	10008,34	9347,84	660,50
AIC	134600,14	134191,05	409,09	134934,19	134092,56	841,63
R^2	0,9540	0,9584	-0,0044	0,9511	0,9574	-0,0063

Tabla 8.3: Indicativos de los diferentes modelos de RNA con VE

Los modelos SVM probados presentaban unos resultados notablemente peores que el resto de redes neuronales estudiadas por lo que estos no se han incluido en los resultados.

Tras analizar los resultados obtenidos, puede apreciarse que el modelo que mejor se ajusta al término de potencia a predecir es el modelo MLP con 19 neuronas en la primera capa oculta, pues es el que mejores resultados presenta en términos de RMSE, MAE y AIC.

Las columnas "diferencia" presentan la resta entre los términos RMSE, MAE y AIC para el estudio con 5-13 neuronas frente al estudio con 14-20 neuronas en la primera capa oculta, pudiéndose apreciar que los modelos con un número de neuronas mayor en la primera capa oculta presentan mejores resultados que con un número de neuronas menor.

En las *figuras 8.5 y 8.6* se pueden observar dos representaciones diferentes de las potencias reales frente a las predichas mediante una red neuronal de tipo GFFN de 1 capa oculta con 14 neuronas en esa primera capa.

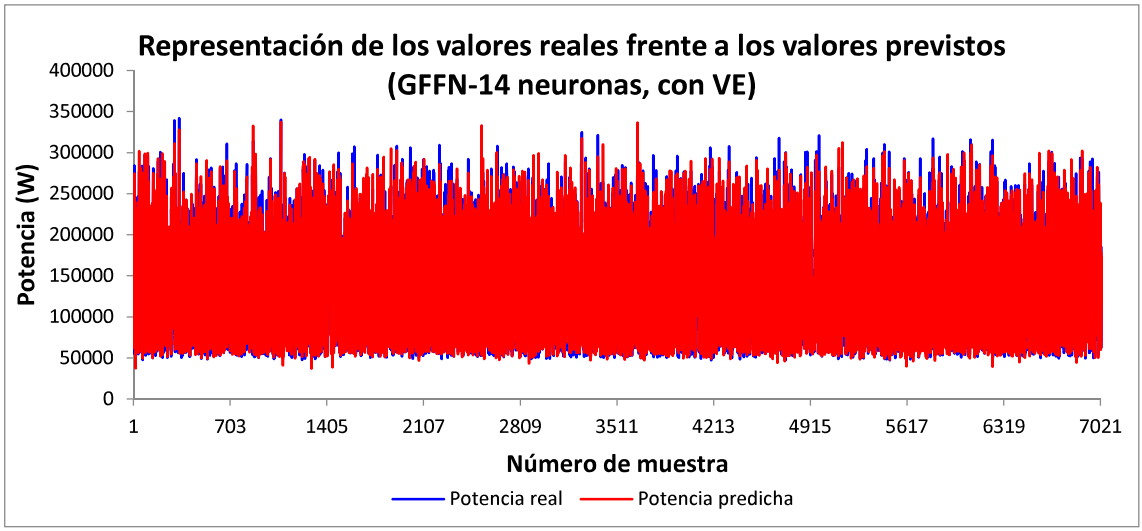

Figura 8.5: Representación de los valores reales de potencia frente a los valores previstos en con un modelo GFFN con 14 neuronas en su primera capa oculta (Con VE) I.

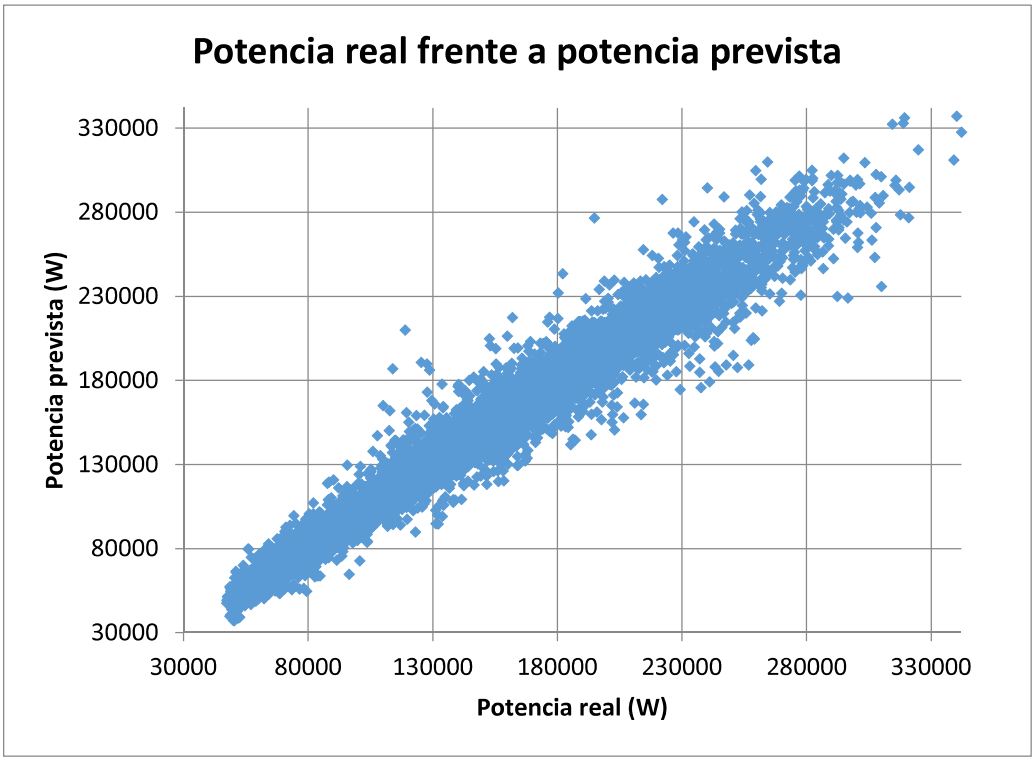

Figura 8.6: Representación de los valores reales de potencia frente a los valores previstos en con un modelo GFFN con 14 neuronas en su primera capa oculta (Con VE) II.

8.3 Modelos creados con Matlab

8.3.1 Sin VE

Dado que se podrán obtener tantos modelos diferentes como valores de radio se prueben a la hora de realizar las agrupaciones (clusters), se han realizado 10 modelos con radios comprendidos desde 1 hasta 0,4 obteniéndose los siguientes resultados:

Sin VE								
Modelos	Radio	Clusters	RMSE			MAE		
			Train	Cross	Test	Train	Cross	Test
Modelo 1	1	5	14254,11	15072,58	15473,90	9535,63	9715,70	9944,73
Modelo 2	0,9	5	14096,69	14612,67	15144,83	9517,55	9632,01	9874,96
Modelo 3	0,8	8	13790,04	14251,07	15090,77	9303,84	9403,97	9763,06
Modelo 4	0,7	9	13499,82	14193,56	14607,46	9129,93	9307,04	9412,63
Modelo 5	0,675	9	13476,11	14122,76	14577,09	9130,22	9248,47	9418,86
Modelo 6	0,65	9	13480,53	14135,49	14607,69	9132,56	9256,25	9443,61
Modelo 7	0,625	9	13450,10	14113,89	14610,61	9138,54	9271,89	9472,62
Modelo 8	0,6	9	13448,41	14183,34	14639,86	9166,00	9348,12	9531,27
Modelo 9	0,5	11	13700,84	14403,55	14896,96	9235,92	9402,60	9611,61
Modelo 10	0,4	12	13598,88	14294,10	14773,13	9164,74	9328,26	9490,98

Tabla 8.4: Indicativos de los diferentes modelos de lógica difusa sin VE.

En las *figuras 8.7 y 8.*8 se representan las evoluciones de los errores MAE y RMSE en el grupo de test en los diferentes modelos estudiados.

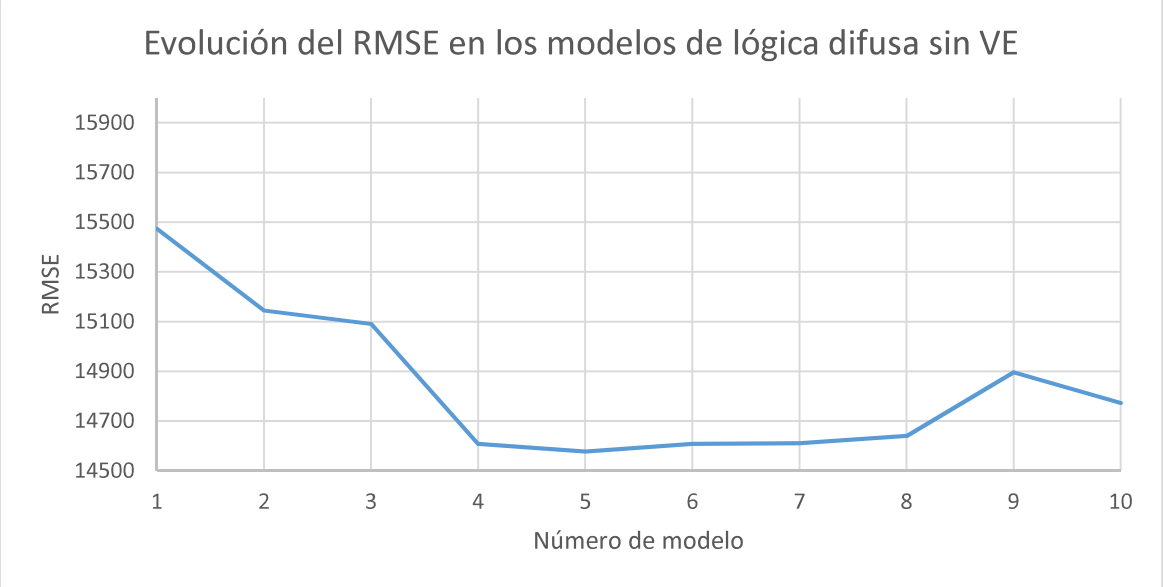

Figura 8.7: Evolución del error RMSE en los modelos de lógica difusa sin VE

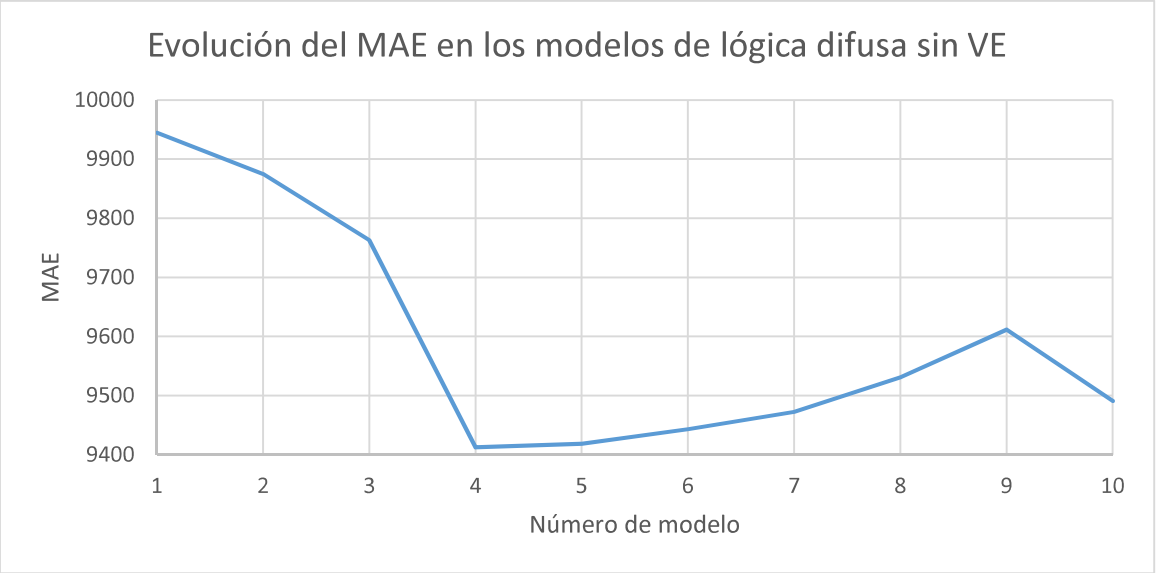

Figura 8.8: Evolución del error MAE en los modelos de lógica difusa sin VE

8.3.2 Con VE

Dado que se podrán obtener tantos modelos diferentes como valores de radio se prueben a la hora de realizar las agrupaciones (clusters), se han realizado 10 modelos con radios comprendidos desde 1 hasta 0,4 obteniéndose los siguientes resultados:

Con VE								
Modelos	**Radio**	**Clusters**	**RMSE**			**MAE**		
			Train	**Cross**	**Test**	**Train**	**Cross**	**Test**
Modelo 1	1	6	14720,77	15149,05	15795,62	9847,66	10077,53	10152,70
Modelo 2	0,9	9	13521,28	14403,10	14902,70	9184,38	9482,19	9685,83
Modelo 3	0,8	9	13564,30	14420,71	14890,29	9202,83	9536,20	9644,82
Modelo 4	0,775	9	13697,43	14451,56	14776,65	9215,12	9547,08	9508,92
Modelo 5	0,75	9	13712,51	14451,87	14779,20	9224,51	9551,80	9513,90
Modelo 6	0,725	9	13737,77	14445,40	14782,68	9255,27	9570,85	9550,28
Modelo 7	0,7	9	13568,24	14148,94	14737,42	9198,62	9443,47	9530,58
Modelo 8	0,6	7	14135,73	14838,51	15038,70	9609,60	9853,10	9823,94
Modelo 9	0,5	7	14400,00	15006,55	15218,24	9762,94	9976,33	9958,18
Modelo 10	0,4	9	14440,62	15168,52	15301,50	9660,88	9875,70	10095,88

Tabla 8.5: Indicativos de los diferentes modelos de lógica difusa con VE.

En las *figuras 8.9 y 8.*10 se representan las evoluciones de los errores MAE y RMSE en los grupos de test en los diferentes modelos estudiados.

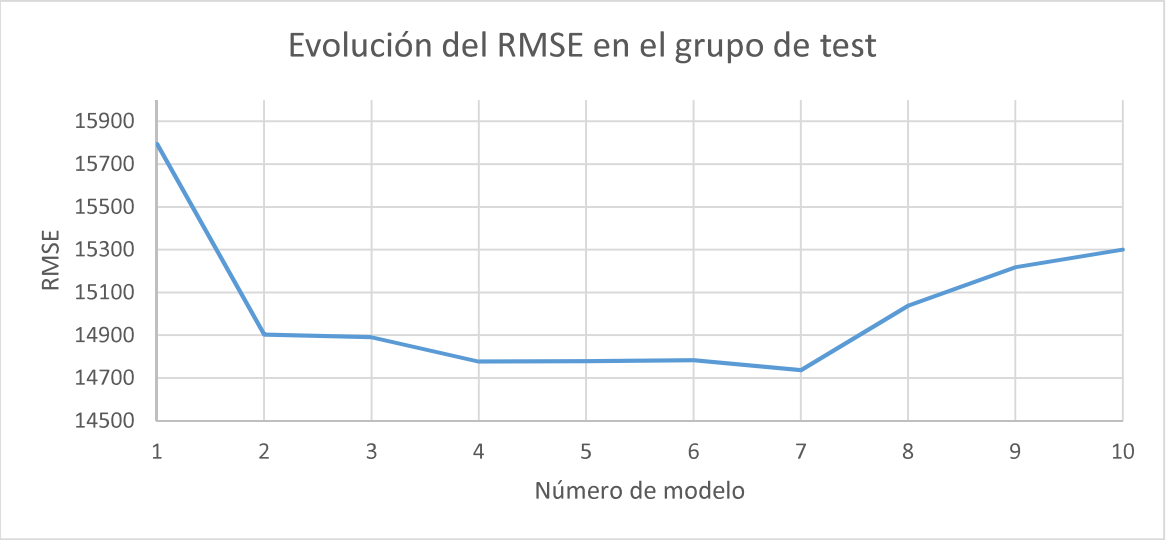

Figura 8.9: Evolución del error RMSE en los modelos de lógica difusa con VE

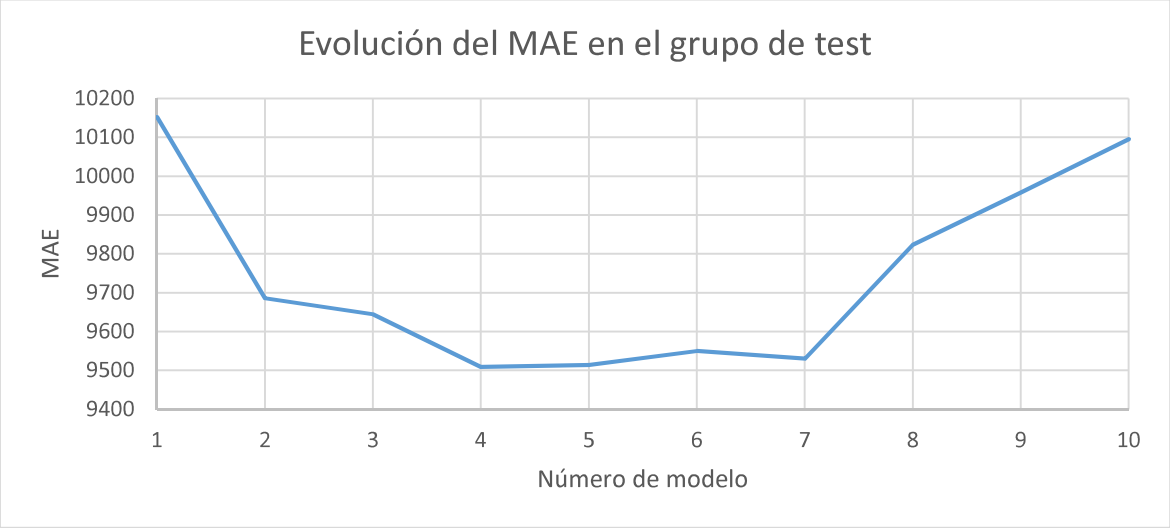

Figura 8.10: Evolución del error MAE en los modelos de lógica difusa con VE

8.3.3 Comparativa entre ambos modelos

Para facilitar la comparativa entre los dos modelos estudiados se han representado los resultados de ambos en conjunto en las *figuras 8.11 y 8.12.*

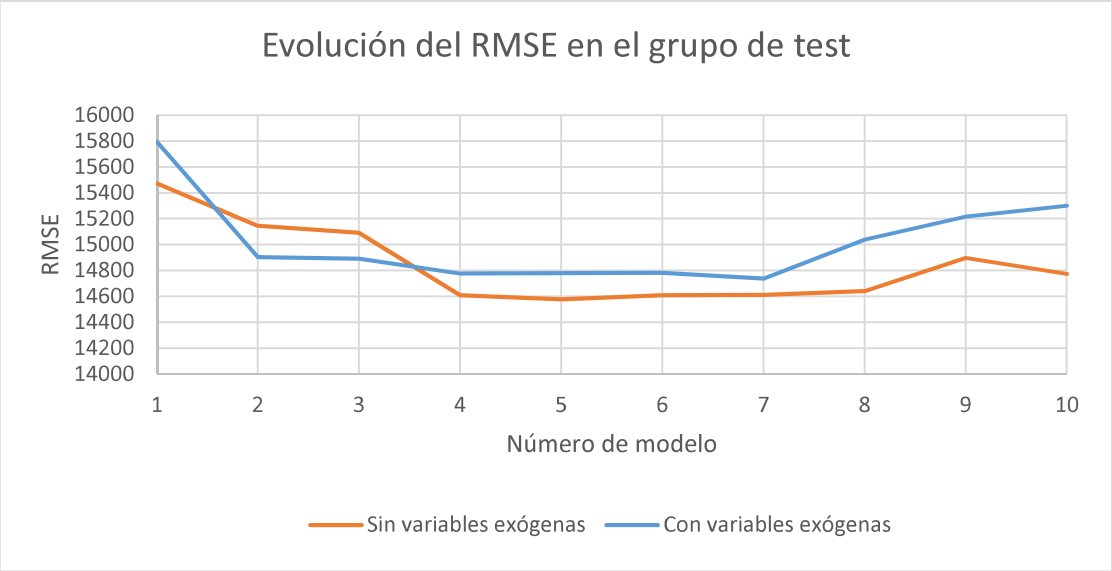

Figura 8.11: Comparativa del error RMSE entre los dos tipos de modelos basados en lógica difusa

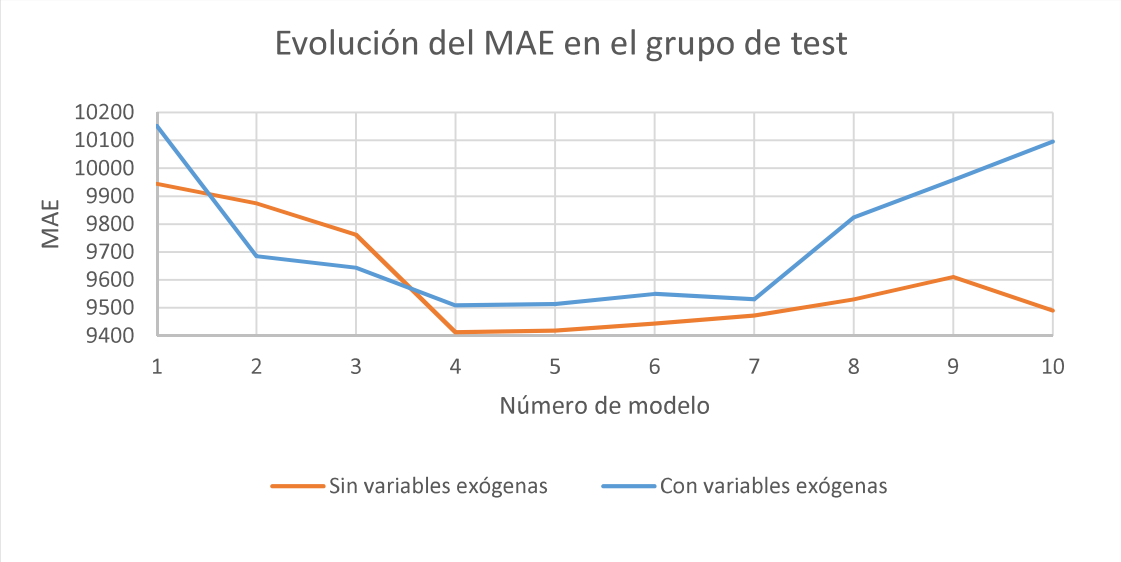

Figura 8.12: Comparativa del error MAE entre los dos tipos de modelos basados en lógica difusa

8.4 Modelos creados con RStudio

8.4.1 Sin VE

8.4.1.1 Modelos de regresión lineal

Los modelos de regresión lineal utilizados y sus resultados en términos de MAE RMSE y R^2 son los presentes en las *tablas 8.6 y 8.7*.

	Lm			Blasso			Blassoaveraged		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	12824	18637	0.9130	12807	18639	0.9130	12808	18639	0.9130
Test	13724	19978	0.9223	13721	20000	0.9223	13723	19998	0.9223

Tabla 8.6: Indicativos de los diferentes modelos de regresión lineal con R (sin VE) I.

	Bridge			Glm.nb			Leapbackward		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	12810	18639	0.9130	12506	19663	0.9130	12506	19663	0.9129
Test	13722	19995	0.2234	13724	21200	0.9126	13362	21200	0.9125

Tabla 8.7: Indicativos de los diferentes modelos de regresión lineal con R (sin VE) II.

Para facilitar la comparativa entre los modelos estudiados se han representado los resultados de éstos en la *figura 8.13*.

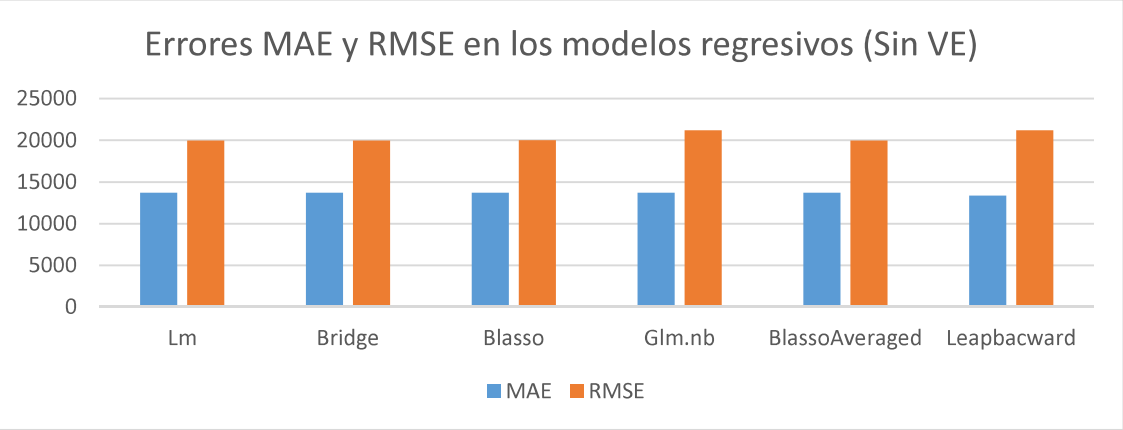

Figura 8.13: Representación de los errores MAE y RMSE en los modelos regresivos (Sin VE)

En la *figura 8.14* se puede ver una representación gráfica del valor real de potencia frente al valor previsto mediante el algoritmo "LeapBackward". La línea roja representaría una predicción perfecta, por lo que, cuanto más cerca de ella estén los puntos mejor se ajusta el modelo.

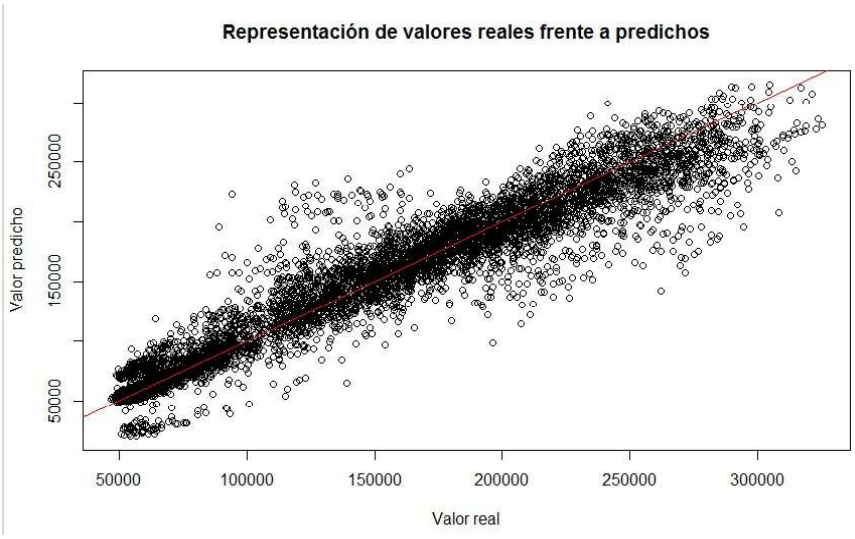

Figura 8.14: Representación de las potencias reales frente a las previstas mediante el algoritmo "LeapBackward" en el grupo de test

8.4.1.2 Modelos de redes neuronales

Los modelos de redes neuronales utilizados y sus resultados en términos de MAE RMSE y R^2 son los presentes en las *tablas 8.8 y 8.9.*

	Brnn			Pcr		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	10400	15152	0.9130	17771	23969	0.9130
Test	10728	16051	0.9499	18444	25029	0.8798

Tabla 8.8: Indicativos de los diferentes modelos de redes neuronales con R (sin VE) I.

	SvmRadialSigma			SvmRadialCost		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	7614	11059	0.9669	7908	11519	0.9130
Test	8539	13057	0.9694	8529	13033	0.9671

Tabla 8.9: Indicativos de los diferentes modelos de redes neuronales con R (sin VE) II.

Para facilitar la comparativa entre los modelos estudiados se han representado los resultados de éstos en la *figura 8.15*.

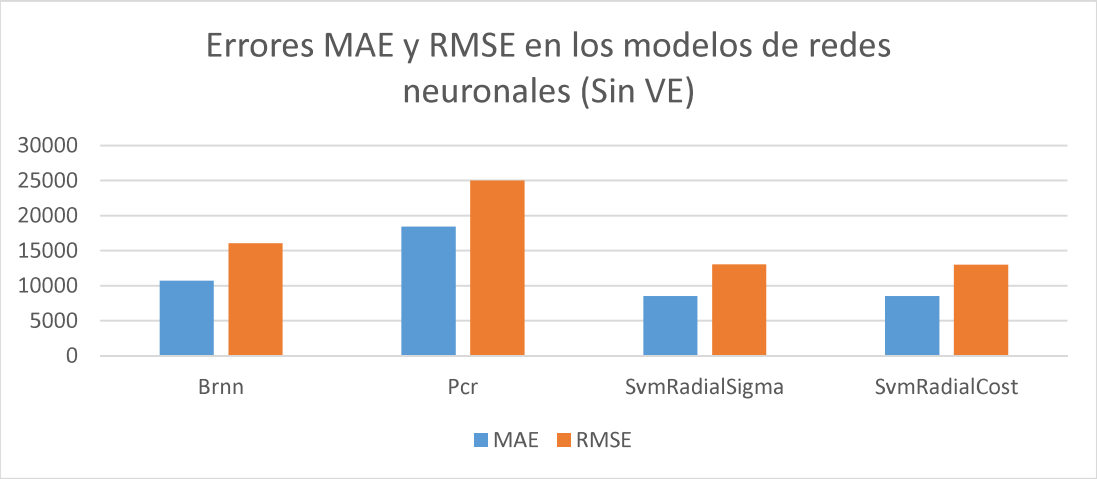

Figura 8.15: Representación de los errores MAE y RMSE en los modelos de redes neuronales (Sin VE)

En la *figura 8.16* se puede ver una representación gráfica del valor real de potencia frente al valor previsto mediante el algoritmo "Brnn". Puede apreciarse que para una potencia superior a 200.000 W el valor real es notablemente superior al valor previsto para muchos de los casos.

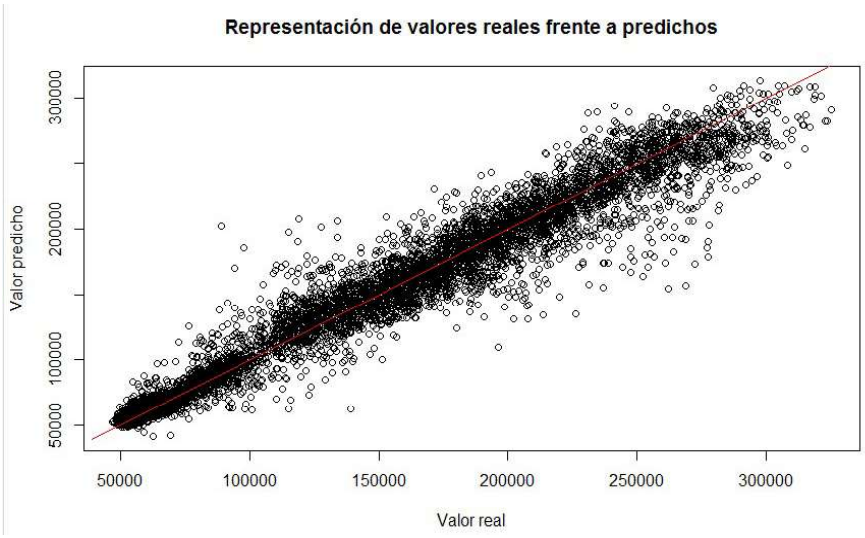

Figura 8.16: Representación de las potencias reales frente a las previstas mediante el algoritmo "Brnn" en el grupo de test

8.4.1.3 Modelos de bosques aleatorios

Los modelos de bosques aleatorios utilizados y sus resultados en términos de MAE RMSE y R^2 son los presentes en las *tablas 8.10 y 8.11.*

	Cubist			Ranger		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	6206	8809	0.9130	4318	6274	0.9130
Test	8692	13315	0.9657	8384	12869	0.9680

Tabla 8.10: Indicativos de los diferentes modelos de bosques aleatorios con R (sin VE) I.

	Rf			Rborist		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	3488	5174	0.9934	3130	4617	0.9679
Test	8774	13496	0.9645	8822	13571	0.9129

Tabla 8.11: Indicativos de los diferentes modelos de bosques aleatorios con R (sin VE) II.

Para facilitar la comparativa entre los modelos estudiados se han representado los resultados de éstos en la *figura 8.17*.

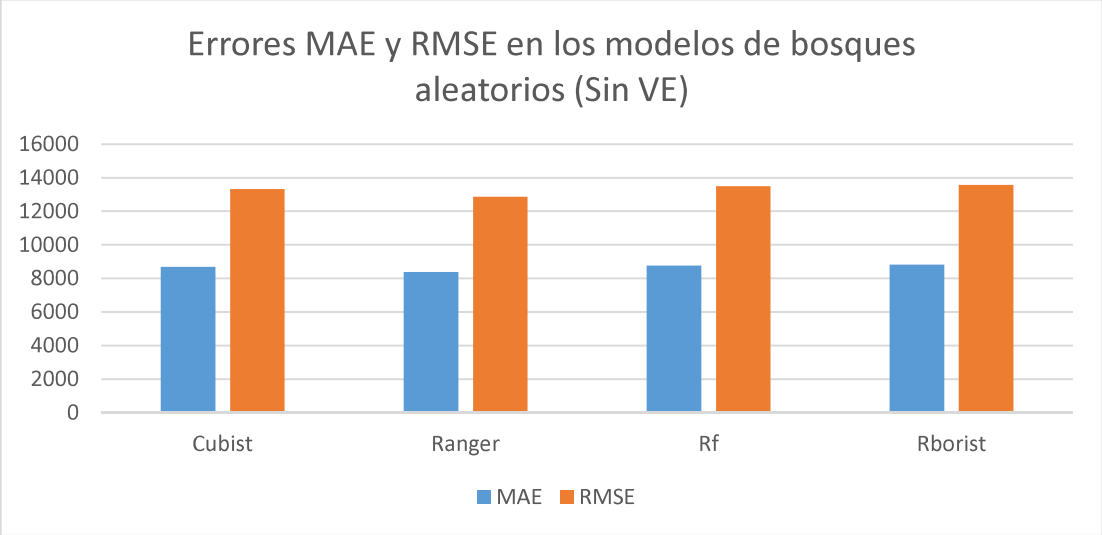

Figura 8.17: Representación de los errores MAE y RMSE en los modelos de bosques aleatorios (Sin VE)

En las *figuras 8.18 y 8.19* se pueden ver dos representaciones gráficas del valor real de potencia frente al valor previsto mediante el algoritmo "Ranger" para los grupos de entrenamiento y test. Puede apreciarse que el ajuste del modelo en el grupo de datos de entrenamiento es notablemente mejor que el ajuste en el grupo de test dado que este modelo de bosques aleatorios tiende a sobreentrenar.

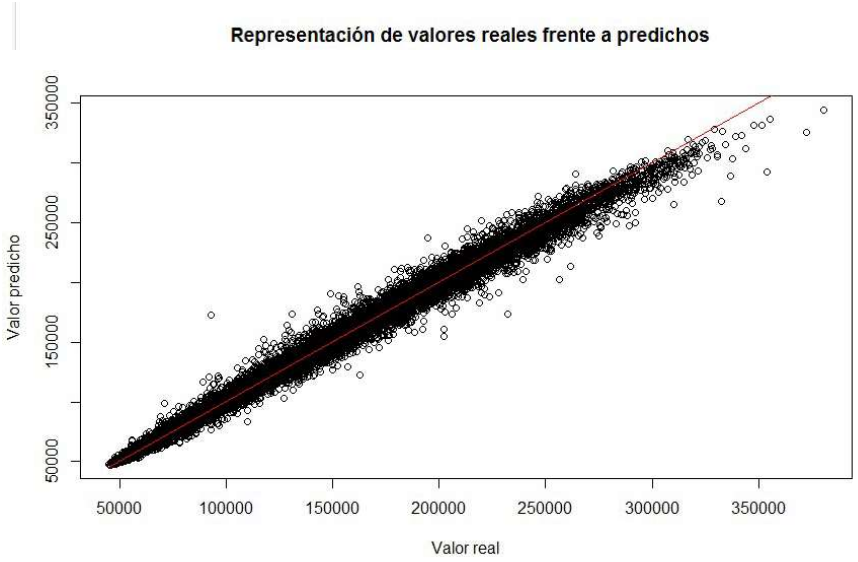

Figura 8.18: Representación de las potencias reales frente a las previstas mediante el algoritmo "Ranger" en el grupo de entrenamiento

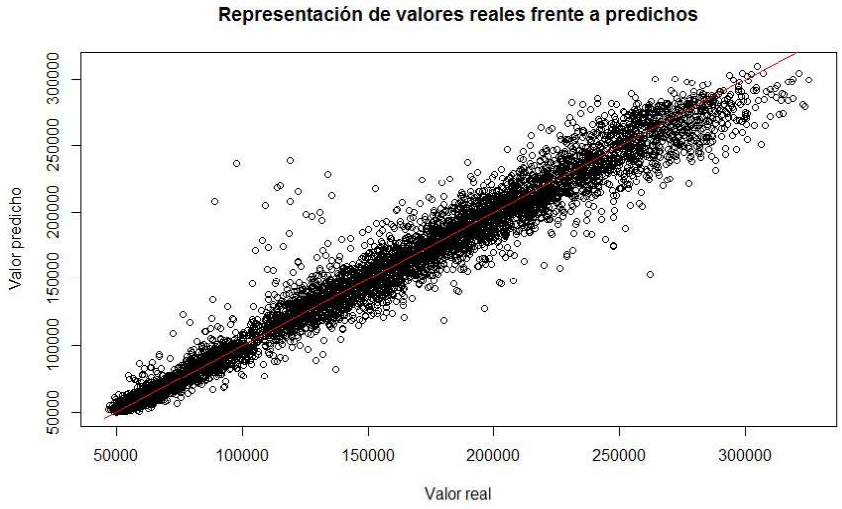

Figura 8.19: Representación de las potencias reales frente a las previstas mediante el algoritmo "Ranger" en el grupo de test

8.4.2 Con VE

8.4.2.1 Modelos de regresión lineal

Los modelos de regresión lineal utilizados y sus resultados en términos de MAE RMSE y R^2 son los presentes en las *tablas 8.12 y 8.13*.

	Lm			Blasso			BlassoAveraged		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	12833	18639	0.9130	12827	18641	0.9130	12816	18641	0.9130
Test	13746	19993	0.9223	13749	20002	0.9223	13744	20007	0.9223

Tabla 8.12: Indicativos de los diferentes modelos de regresión lineal con R (con VE) I.

	Bridge			Glm.nb			Leapbackward		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	12811	18642	0.9130	12507	19665	0.9032	12755	19135	0.9083
Test	13732	20005	0.9223	13371	21224	0.9125	13743	20644	0.9171

Tabla 8.13: Indicativos de los diferentes modelos de regresión lineal con R (con VE) II.

Para facilitar la comparativa entre los modelos estudiados se han representado los resultados de éstos en la *figura 8.20.*

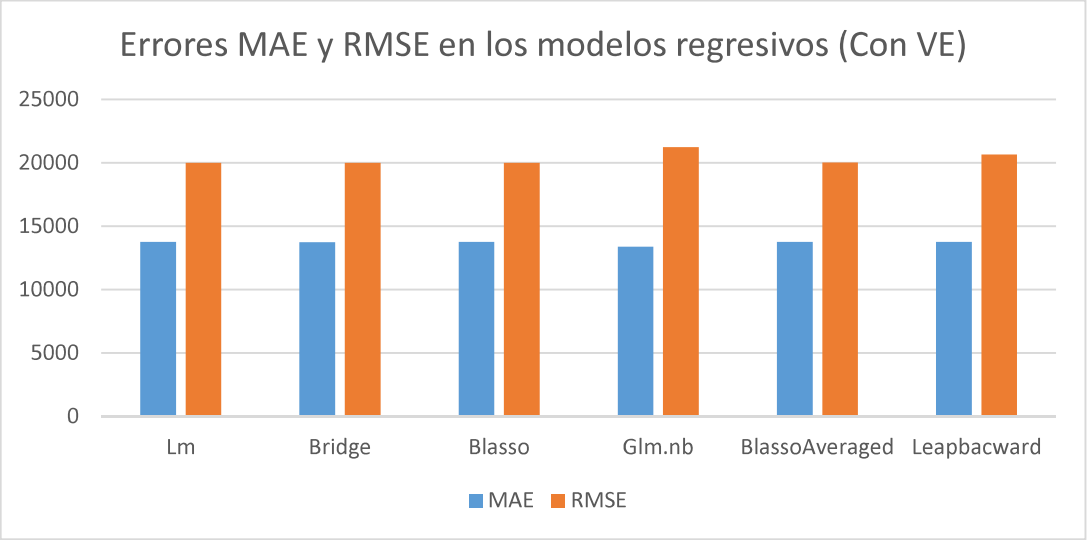

Figura 8.20: Representación de los errores MAE y RMSE en los modelos regresivos (Con VE)

En la *figura 8.21* se puede ver una representación gráfica del valor real de potencia frente al valor previsto mediante el algoritmo "Lm". Puede apreciarse que para una potencia superior a 200.000 W el valor real es notablemente superior al valor previsto para muchos de los casos.

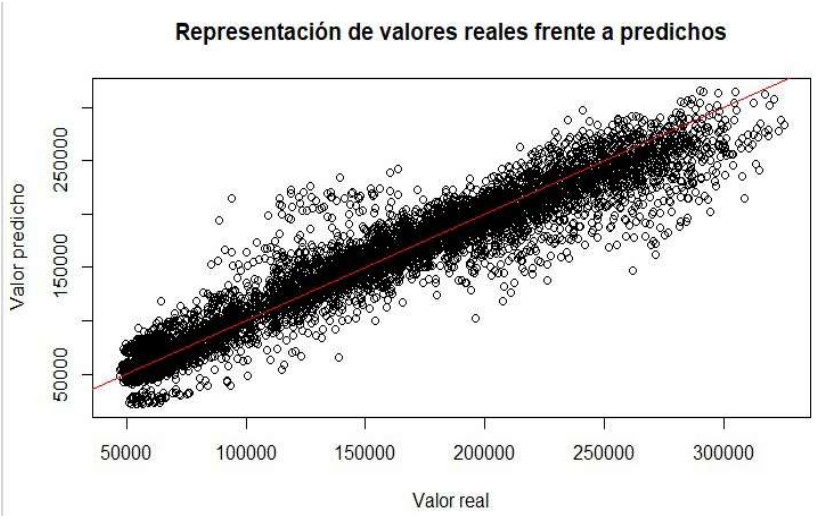

Figura 8.21: Representación de las potencias reales frente a las predichas mediante el algoritmo "Lm" en el grupo de test

8.4.2.2 Modelos de redes neuronales

Los modelos de redes neuronales utilizados y sus resultados en términos de MAE RMSE y R^2 son los presentes en las tablas *8.14 y 8.15.*

	Brnn			Pcr		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	9420	14069	0.9504	19061	26614	0.8226
Test	9594	14355	0.9600	19565	27352	0.8547

Tabla 8.14: Indicativos de los diferentes modelos de redes neuronales con R (con VE) I.

	SvmRadialCost			SvmRadialSigma		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	7650	11049	0.9695	7286	10486	0.9504
Test	8328	12483	0.9697	8376	12540	0.9600

Tabla 8.15: Indicativos de los diferentes modelos de redes neuronales con R (con VE) II.

Para facilitar la comparativa entre los modelos estudiados se han representado los resultados de éstos en la *figura 8.22*.

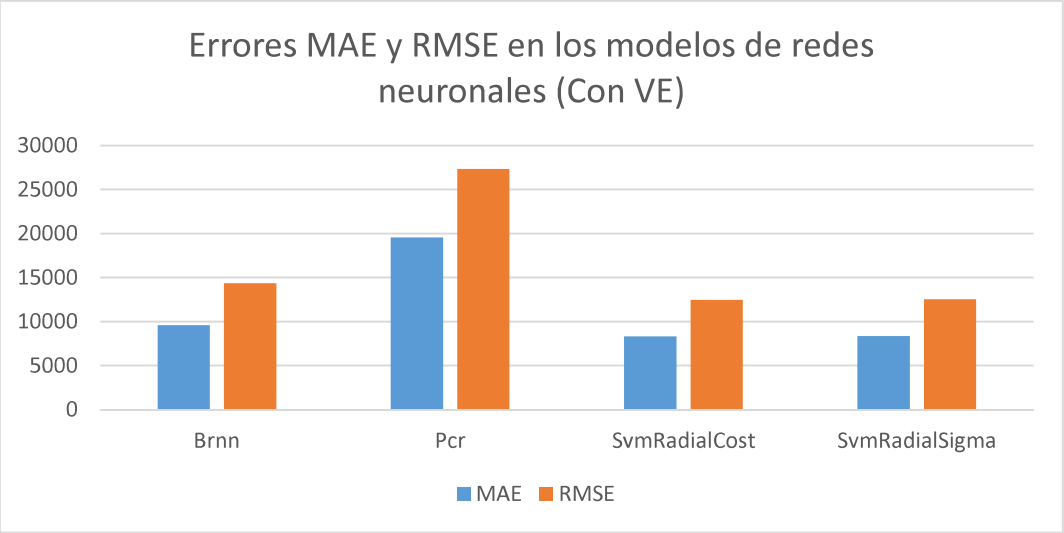

Figura 8.22: Representación de los errores MAE y RMSE en los modelos de redes neuronales (Con VE)

En la *figura 8.23* se puede ver una representación gráfica del valor real de potencia frente al valor previsto mediante el algoritmo "SvmRadialCost". Puede apreciarse que para una potencia superior a 200.000 W el valor real es notablemente superior al valor previsto para muchos de los casos.

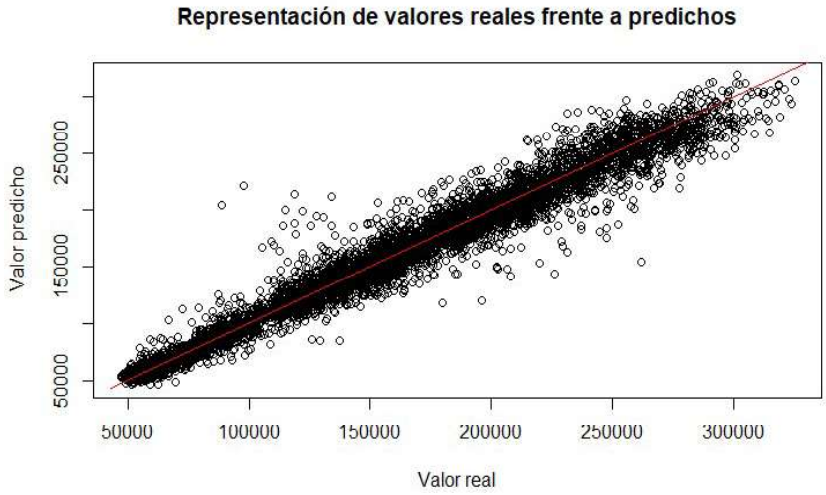

Figura 8.23: Representación de las potencias reales frente a las predichas mediante el algoritmo "SvmRadialCost" en el grupo de test

8.4.2.3 Modelos de bosques aleatorios

Los modelos de bosques aleatorios utilizados y sus resultados en términos de MAE RMSE y R^2 son los presentes en las *tablas 8.16 y 8.17*.

	Cubist			Ranger		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	6144	8659	0.9812	4055	5866	0.9915
Test	8502	12963	0.9675	8189	12462	0.9699

Tabla 8.16: Indicativos de los diferentes modelos de bosques aleatorios con R (con VE) I.

	Rf			Rborist		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	3366	4978	0.9939	3072	4507	0.9950
Test	8599	13107	0.9667	8651	13236	0.9660

Tabla 8.17: Indicativos de los diferentes modelos de bosques aleatorios con R (con VE) II.

Para facilitar la comparativa entre los modelos estudiados se han representado los resultados de éstos en la *figura 8.24*.

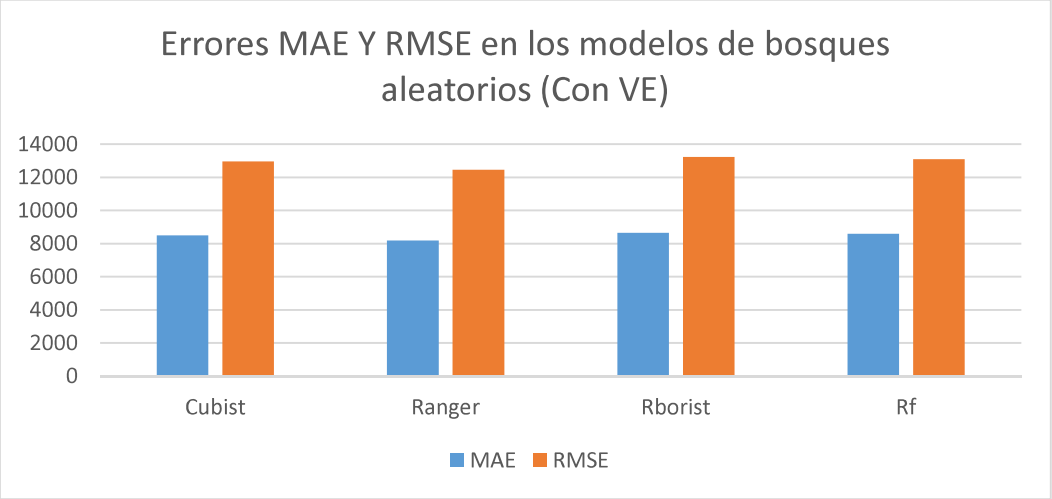

Figura 8.24: Representación de los errores MAE y RMSE en los modelos de bosques aleatorios (Con VE)

En la *figura 8.25* se puede ver una representación gráfica del valor real de potencia frente al valor previsto mediante el algoritmo "Cubist".

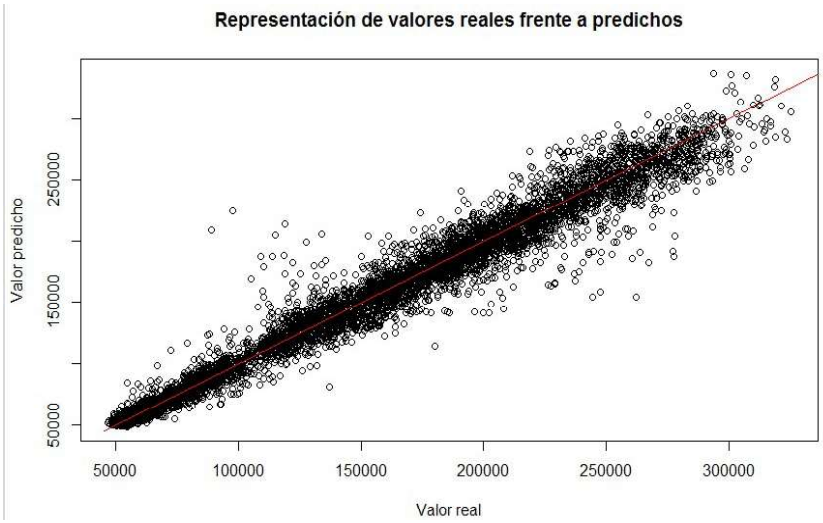

Figura 8.25: Representación de las potencias reales frente a las predichas mediante el algoritmo "Cubist" en el grupo de test

8.4.3 Aplicación de la poda en los modelos de bosques aleatorios

En las *tablas 8.18 y 8.19* se reflejan los resultados obtenidos tras la aplicación de la poda con el agoritmo "rf".

	Rf		Rf	
	Min.node=10	mtry=6	Min.node=20	mtry=6
	MAE	RMSE	MAE	RMSE
Entrenamiento	4145	6106	5231	7602
Test	8612	13127	8652	13174

Tabla 8.18: Comparativa de diferentes modelos creados con el algoritmo "rf" variando el tamaño mínimo de nodo. I

	Rf		Rf	
	Min.node=30	mtry=6	Min.node=40	mtry=6
	MAE	RMSE	MAE	RMSE
Entrenamiento	5899	8520	6373	9245
Test	8687	13245	8764	13413

Tabla 8.19: Comparativa de diferentes modelos creados con el algoritmo "rf" variando el tamaño mínimo de nodo. II

En la *figura 8.26* se aprecia claramente como conforme aumenta el tamaño mínimo de nodo, aumenta notablemente el error MAE en el grupo de entrenamiento(azul) y levemente en el grupo de test (naranja).

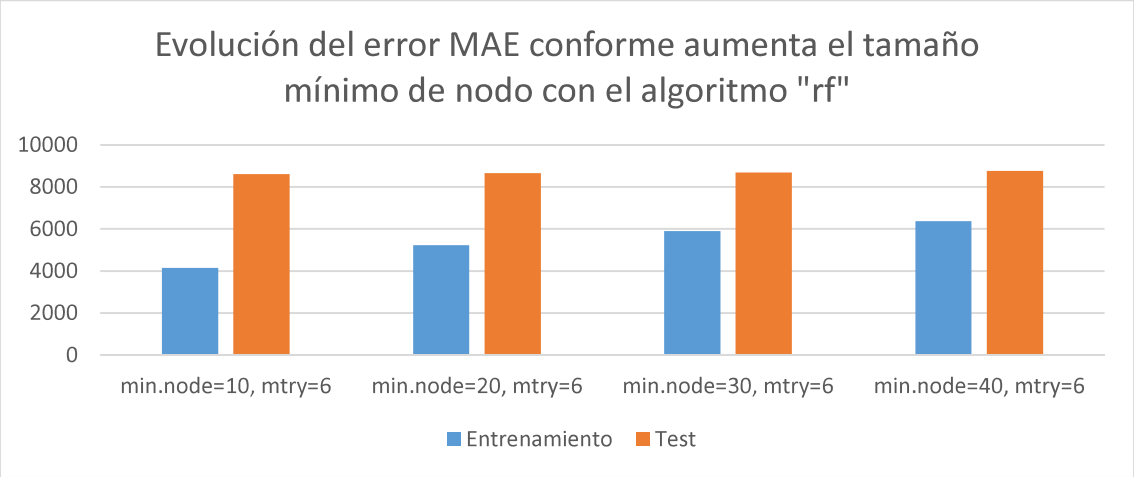

Figura 8.26: Evolución del error MAE conforme aumenta el tamaño mínimo de nodo con el algoritmo "rf".

En las *tablas 8.20 y 8.21* se reflejan los resultados obtenidos tras la aplicación de la poda con el agoritmo "rf".

	Ranger		Ranger	
	Min.node=5	mtry=4	Min.node=10	mtry=4
	MAE	RMSE	MAE	RMSE
Entrenamiento	3649	5376	4456	6506
Test	8494	12845	8513	12898

Tabla 8.20: Comparativa de diferentes modelos creados con el algoritmo "ranger" variando el tamaño mínimo de nodo. I

	Ranger		Ranger	
	Min.node=20	mtry=6	Min.node=30	mtry=6
	MAE	RMSE	MAE	RMSE
Entrenamiento	5283	7662	5913	8554
Test	8631	13105	8651	13174

Tabla 8.21: Comparativa de diferentes modelos creados con el algoritmo "ranger" variando el tamaño mínimo de nodo. II

En la *figura 8.27* se aprecia claramente como conforme aumenta el tamaño mínimo de nodo, aumenta notablemente el error MAE en el grupo de entrenamiento (azul) y levemente en el grupo de test (naranja).

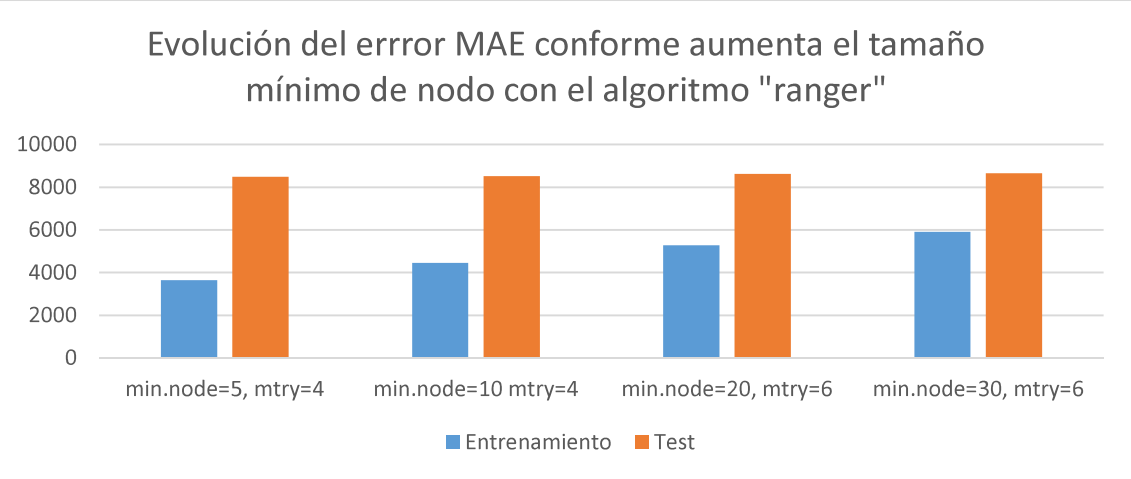

Figura 8.27: Evolución del error MAE conforme aumenta el tamaño mínimo de nodo con el algoritmo "ranger".

8.4.4 Modificación de los modelos de bosques aleatorios

Los resultados obtenidos del análisis mediante el mallado explicado en el *subapartado 7.6.5* son los reflejados en la *tabla 8.22.*

	Ranger por defecto			Ranger modificado		
	MAE	RMSE	R^2	MAE	RMSE	R^2
Entrenamiento	4055	5866	0.9915	3613	5273	0.9686
Test	8189	12462	0.9699	8403	12729	0.9932

Tabla 8.22: Comparativa entre el modelo "ranger" por defecto y el modificado.

Puede comprobarse que el modelo modificado mejora los resultados en el grupo de entrenamiento, pero los empeora en el grupo de test.

8.5 Tabla resumen de todos los modelos evaluados

En la *tabla 8.23* se muestran todos los modelos evaluados ordenados de menor a mayor error MAE a modo de poder realizar una comparativa global de todo el estudio.

Modelos	Variables exógenas	Programa informático	Tipo de modelo	Características de los modelos	MAE	RMSE
MLP	Sin VE	Neurosolutions	Redes neuronales	13 neuronas	8183.1	11796.6
Ranger	Con VE	R Studio	Bosques aleatorios	500 árboles	8188.8	12462.4
MLP	Con VE	Neurosolutions	Redes neuronales	19 neuronas	8249.5	11777.1
MLP	Sin VE	Neurosolutions	Redes neuronales	20 neuronas	8276.7	11931.4
GFFN	Sin VE	Neurosolutions	Redes neuronales	19 neuronas	8309.0	12028.7
SvmRadialCost	Con VE	R Studio	Redes neuronales		8328.1	12482.8
GFFN	Con VE	Neurosolutions	Redes neuronales	14 neuronas	8338.4	11860.0
MLP	Con VE	Neurosolutions	Redes neuronales	13 neuronas	8341.8	11867.9
SvmRadialSigma	Con VE	R Studio	Redes neuronales		8375.8	12539.9
Ranger	Sin VE	R Studio	Bosques aleatorios	500 árboles	8384.3	12869.2
GFFN	Sin VE	Neurosolutions	Redes neuronales	10 neuronas	8455.7	12331.9
Cubist	Con VE	R Studio	Bosques aleatorios	500 árboles	8502.2	12962.5
GFFN	Con VE	Neurosolutions	Redes neuronales	12 neuronas	8514.1	12083.9
SvmRadialCost	Sin VE	R Studio	Redes neuronales		8529.4	13032.7
SvmRadialSigma	Sin VE	R Studio	Redes neuronales		8539.5	13057.5
Rf	Con VE	R Studio	Bosques aleatorios	500 árboles	8599.2	13106.7
Rborist	Con VE	R Studio	Bosques aleatorios	500 árboles	8651.4	13235.8
Cubist	Sin VE	R Studio	Bosques aleatorios	500 árboles	8692.3	13314.9

Modelos	Variables exógenas	Programa informático	Tipo de modelo	Características de los modelos	MAE	RMSE
Rf	Sin VE	R Studio	Bosques aleatorios	500 árboles	8774.3	13496.5
PCA	Sin VE	Neurosolutions	Redes neuronales	17 neuronas	8788.0	13151.7
Rborist	Sin VE	R Studio	Bosques aleatorios	500 árboles	8822.2	13571.0
PCA	Sin VE	Neurosolutions	Redes neuronales	11 neuronas	8827.2	13187.2
RBF	Con VE	Neurosolutions	Redes neuronales	16 neuronas	9347.8	13515.2
RBF	Sin VE	Neurosolutions	Redes neuronales	12 neuronas	9356.6	13766.7
RBF	Sin VE	Neurosolutions	Redes neuronales	14 neuronas	9397.3	13635.3
Mod. 4 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,7, clusters=9	9412.6	14607.5
Mod. 5 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,675, clusters=9	9418.9	14577.1
Mod. 6 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,65, clusters=9	9443.6	14607.7
Mod. 7 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,625, clusters=9	9472.6	14610.6
PCA	Con VE	Neurosolutions	Redes neuronales	15 neuronas	9472.9	13635.1
Mod. 10 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,4, clusters=12	9491.0	14773.1
Mod. 4 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,775, clusters=9	9508.9	14776.7
Mod. 5 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,75, clusters=9	9513.9	14779.2
Mod. 7 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,7, clusters=9	9530.6	14737.4
Mod. 8 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,6, clusters=9	9531.3	14639.9
Mod. 6 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,725, clusters=9	9550.3	14782.7
Brnn	Con VE	R Studio	Redes neuronales	Iteraciones= 1000	9594.1	14355.3
Mod. 9 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,5, clusters=11	9611.6	14897.0
Mod. 3 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,8, clusters=9	9644.8	14890.3

Modelos	Variables exógenas	Programa informático	Tipo de modelo	Características de los modelos	MAE	RMSE
Mod. 2 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,9, clusters=9	9685.8	14902.7
Mod. 3 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,9, clusters=8	9763.1	15090.8
Mod. 8 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,6, clusters=7	9823.9	15038.7
PCA	Con VE	Neurosolutions	Redes neuronales	13 neuronas	9830.7	14086.6
Mod. 2 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=0,9, clusters=5	9875.0	15144.8
Mod. 1 logica difusa	Sin VE	Matlab	Lógica difusa	Radio=1, clusters=5	9944.7	15473.9
Mod. 9 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,5, clusters=7	9958.2	15218.2
RBF	Con VE	Neurosolutions	Redes neuronales	11 neuronas	10008.3	14474.4
Mod. 10 logica difusa	Con VE	Matlab	Lógica difusa	Radio=0,4 clusters=9	10095.9	15301.5
Mod. 1 logica difusa	Con VE	Matlab	Lógica difusa	Radio=1, clusters=6	10152.7	15795.6
Brnn	Sin VE	R Studio	Redes neuronales	Iteraciones = 1000	10727.6	16050.8
Regresivo V3	Sin VE	Excel	Regresivo		13299.1	22162.7
Regresivo V6	Con VE	Excel	Regresivo		13336.5	22205.9
Regresivo V5	Con VE	Excel	Regresivo		13338.4	22208.5
Leapbackward	Sin VE	R Studio	Regresivo		13362.4	21200.3
Glm.nb	Con VE	R Studio	Regresivo		13371.4	21224.2
Regresivo V4	Sin VE	Excel	Regresivo		13470.4	22317.1
Blasso	Sin VE	R Studio	Regresivo		13721.4	19999.7
Bridge	Sin VE	R Studio	Regresivo		13722.0	19995.2
Blassoaveraged	Sin VE	R Studio	Regresivo		13722.9	19998.0
Lm	Sin VE	R Studio	Regresivo		13724.4	19978.5

Modelos	Variables exógenas	Programa informático	Tipo de modelo	Características de los modelos	MAE	RMSE
Glm.nb	Sin VE	R Studio	Regresivo		13724.4	21200.3
Bridge	Con VE	R Studio	Regresivo		13732.2	20005.4
Leapbackward	Con VE	R Studio	Regresivo		13742.6	20643.6
BlassoAveraged	Con VE	R Studio	Regresivo		13743.8	20006.8
Lm	Con VE	R Studio	Regresivo		13745.9	19992.6
Blasso	Con VE	R Studio	Regresivo		13748.9	20002.2
Día similar	Sin VE	Excel	Persistente		14184.5	24336.5
Pcr	Sin VE	R Studio	Redes neuronales		18444.3	25028.8
Pcr	Con VE	R Studio	Redes neuronales		19564.8	27352.4
Regresivo V2	Sin VE	Excel	Regresivo		20745.5	35510.7
Regresivo V2	Sin VE	Excel	Regresivo		20836.5	35736.7
Día anterior	Sin VE	Excel	Persistente		21419.0	36411.8

Tabla 8.23: Comparativa de todos los modelos estudiados

9. Conclusiones

- Los modelos realizados no son generalizables a cualquier grupo de consumidores de potencia, pues han sido realizados para un grupo concreto de consumidores.
- De igual modo que los modelos no son generalizables para un grupo cualquiera de consumidores, tampoco lo son para una previsión cualquiera con independencia del año a predecir, pues, los modelos han sido elaborados con datos entre los años 2008 y 2013, y, hay variables que en esos años no son muy trascendentes, como es la radiación solar, variable que, con las facilidades que se presentan en la actualidad para el autoconsumo, será de gran consideración.
- Los modelos que mejores resultados presentan son los basados en RNA creados mediante Neurosolutions y los basados en bosques aleatorios creados con RStudio.
- Los peores resultados se obtienen con los modelos persistentes creados con Excel y los basados en regresión lineal creados con RStudio.

9.1 Modelos creados con Excel

En base a los resultados obtenidos se pueden enunciar las siguientes conclusiones:

- Como puede apreciarse, los modelos lineales tienen sus limitaciones, dado que, llegado un momento, aunque se introduzcan variables nuevas el modelo no mejora.
- En todos los modelos el término fijo de potencia es prácticamente cero, por lo que su influencia es nula.
- Los términos pertenecientes a la potencia de dos días antes y a la potencia mínima son términos que representan porcentajes sobre un término de potencia anterior, por lo que a la vista de sus coeficientes suponen menos de un 1% en términos de potencia.
- Los términos que multiplican a las variables sábado y lunes (cuyo valor es 1) presentan un valor máximo de 155,54 (lunes) que, en comparación con los valores de potencia, que los mínimos son en torno a 50.000 W, que hace que la influencia de estos términos sea casi nula.
- La diferencia entre los errores cometidos en los datos de entrenamiento y los cometidos en los datos de evaluación son muy pequeños, por lo que, la

muestra de entrenamiento elegida es lo suficientemente grande para que el modelo creado se ajuste a la realidad.

9.2 Modelos creados con Neurosolutions

- Los modelos que mejores resultados presentan son de tipo MLP.
- Los modelos de redes neuronales creados con Neurosolutions mejoran notablemente los modelos de regresión lineal creados mediante Excel.
- En general, para el mismo tipo de red, la utilización de más neuronas en la primera capa oculta mejora los resultados obtenidos con un número menor de neuronas, y, en los modelos estudiados, pese a complicarse estos añadiendo mayor cantidad de neuronas en la capa oculta, el AIC resultante es menor, lo que indica que, pese a que el modelo es más complejo, esa complejidad es asumible en términos de mejora del error.

9.2.1 Sin VE

- En todos los modelos excepto en el MLP, la utilización de más neuronas en la primera capa oculta mejora los resultados obtenidos con un número menor de neuronas, siendo a su vez, estos tipos de redes los que mejor resultados arrojan.
- La red que mejores resultados presenta es la de tipo MLP con 11 neuronas en su primera capa oculta, lo que hace pensar que una red con un número superior a 20 neuronas no mejorará el resultado obtenido por esta red.

9.2.2 Con VE

- El modelo que mejor se ajusta a la predicción del término potencia demandada es el modelo MLP con 19 neuronas en su primera capa oculta. Por lo que es posible que un modelo con un número de neuronas superior a 20 presente unos resultados mejores que el modelo de 19 neuronas.
- En todos los modelos excepto en el MLP, en el estudio con un rango de 14-20 neuronas el mejor resultado no es próximo a 20 neuronas, por lo que puede concluirse que, aun aumentando por encima de 20 el número de neuronas el modelo resultante no mejorará significativamente los resultados.

9.3 Modelos basados con Matlab

- Los modelos basados en lógica difusa presentan peores resultados que los modelos creados con Neurosolutions y mejores que los creados mediante Excel.
- El número de agrupaciones (clusters) óptimo se determina teóricamente como la raíz cuadrada de las entradas, pero, tras analizar los resultados se aprecia que el número óptimo de agrupaciones óptimo es mayor.
- Los modelos sin VE arrojan mejores resultados que los que tienen en cuenta las variables exógenas.

9.3.1 Sin VE

- El modelo que mejor comportamiento tiene en términos de RMSE es el modelo 5, y en términos de MAE es el modelo 4, los cuales presentan radios de 0,7 y 0,675 respectivamente, dado que los errores entre ambos distan muy poco se elegirá como mejor modelo el modelo 4 dado que tiene un error MAE inferior, siendo este un indicativo más significativo en los ámbitos de predicciones de datos.
- Como es de esperar, a menor radio utilizado para determinar las agrupaciones, mayor es el número de agrupaciones generadas por el modelo.
- Teóricamente, el número óptimo de agrupaciones (clusters) debería de ser la raíz cuadrada del número de entradas, siendo en este caso raíz de 10, es decir, aproximadamente 3, sin embargo, no hay ningún valor de radio posible (el valor máximo de radio es 1) que haga que las agrupaciones creadas sean 3, y los mejores resultados se obtienen con 9 agrupaciones.
- Los errores obtenidos en las muestras de entrenamiento y validación cruzada son sensiblemente mayores que los obtenidos en la muestra de test, lo que quiere decir que la muestra de entrenamiento y validación cruzada es lo suficientemente grande como para que el modelo creado tenga un buen comportamiento frente a nuevas muestras de test. Del mismo modo, esto significa que el modelo no presenta sobreentrenamiento.

9.3.2 Con VE

- El modelo que mejor comportamiento tiene en términos de RMSE es el modelo 7, y en términos de MAE es el modelo 4, los cuales presentan radios de 0,7 y 0,775 respectivamente, dado que los errores entre ambos distan muy poco se elegirá como mejor modelo el modelo 4 dado que tiene un error MAE inferior, siendo este un indicativo más significativo en los ámbitos de predicciones de datos.
- Conforme se reduce el tamaño del radio el número de agrupaciones (clusters) aumenta excepto en el paso de radio 0,7 a 0,6 en el que el número de agrupaciones disminuye de 9 a 7.
- Teóricamente, el número óptimo de agrupaciones (clusters) debería de ser la raíz cuadrada del número de entradas, siendo en este caso raíz de 12, es decir, entre 3 y 4, sin embargo, no hay ningún valor de radio posible (el valor máximo de radio es 1) que haga que las agrupaciones sean 3 o 4, y los mejores resultados se obtienen con 9 agrupaciones.
- Los errores obtenidos en las muestras de entrenamiento y validación cruzada son sensiblemente mayores que los obtenidos en la muestra de test, lo que quiere decir que la muestra de entrenamiento y validación cruzada es lo suficientemente grande como para que el modelo creado tenga un buen comportamiento frente a nuevas muestras de test. Del mismo modo, esto significa que el modelo no presenta sobreentrenamiento.

9.4 Modelos creados con Rstudio

Las conclusiones obtenidas serán generalizables tanto para los modelos con VE como los modelos sin VE dado que, los resultados obtenidos son muy similares.

- Los modelos basados en Redes Neuronales mediante Rstudio, presentan, por lo general, peores resultados que los obtenidos con Neurosolutions.
- Los modelos regresivos creados mediante RStudio proporcionan mejores resultados que los realizados con Excel.
- Los modelos de bosques aleatorios realizados mediante RStudio presentan sobreentrenamiento dado que, los errores obtenidos en el grupo de entrenamiento son mucho menores que en el grupo de test.

- Tanto los modelos regresivos como los basados en redes neuronales presentan desajustes cuando la potencia es superior a 200.000 W siendo el valor real notablemente superior al valor previsto en muchos de los casos.
- Los modelos basados en bosques aleatorios y los basados en redes neuronales presentan mejores resultados teniendo en cuenta las variables exógenas que sin tenerlas, mientras que los modelos lineales presentan prácticamente resultados idénticos en ambos casos.

9.4.1 Aplicación de la poda en modelos de bosques aleatorios

La realización de una poda basada en limitar el tamaño de nodo final hace que:

- Conforme el tamaño del nodo final aumenta (modelo más generalista), los errores en el grupo de entrenamiento también aumentan.
- Conforme el tamaño del nodo final aumenta (modelo más generalista), los errores en el grupo de test aumentan levemente.

Las dos conclusiones anteriores reflejan que el sobreentrenamiento presente en los bosques aleatorios en el grupo de entrenamiento hace que los modelos presenten buenos resultados, ya que si limitamos ese sobreentrenamiento los resultados en el grupo de test empeoran.

9.4.2 Modelos de bosques aleatorios modificados

Tras analizar los resultados obtenidos con el modelo que abarca un gran número de variables se obtienen las siguientes conclusiones:

- Los errores MAE y RMSE disminuyen en el grupo de entrenamiento y aumentan en el grupo de test respecto al modelo con los parámetros por defecto, lo que quiere decir que el sobreentrenamiento generado es perjudicial.
- El defecto de la metodología utilizada es que para la elección del mejor modelo de todos los evaluados se evalúan los resultados en el grupo de entrenamiento y no en el grupo de test, además de que, el mejor modelo se elige en términos de RMSE y no de MAE.

10. Bibliografía

Libros:

- Witten, I.H., Frank, E. *Data Minning Practical Machine Learning Tools and Techniques* (2nd Edition). Elsevier
- Webb, A. (2002) Statistical Pattern Recognition, Wiley
- Hernández-Orallo, J., Ramírez, M.J., Ferri, C. (2004) *Introducción a la Minería de Datos*, Pearson, Prentice Hall
- Haykin S. (1999). Neural Networks A Comprehensive Foundation. Pearson Education.
- K.-L. Du and M.N.S. Swamy (2006). Neural Networks in a Softcomputing Framework. Springer.
- Baruque B., Corchado E. (2010). *Fusion Methods for Unsupervised Learning Ensembles*. Springer.
- Steinwart , I., Christmann A. (2008) *Support Vector Machines*. Springer
- Pazos A., M. Dans. C. *Integración de las redes de neuronas artificiales con Lógica Difusa*. Universidad de A Coruña
- Ruiz, M. (2007) *Diagram of a typical myelinated vertebrate motoneuron*. , recuperado de http://en.wikipedia.org/wiki/Neuron

Documentos difundidos en internet:

- Alejandro Pazos & César M. Dans (1996) *Integración de las Redes de Neuronas Artificiales con Lógica Difusa.* Recuperado de: https://core.ac.uk/download/pdf/61904994.pdf. (Última consulta 26 de junio de 2019)
- Alfonso Ballesteros. *Neural Networks Framework*. Recuperado de: http://www.redes-neuronales.com.es/tutorial-redes-neuronales/clasificacion-de-las-redes-neuronales-artificiales.htm. (Última consulta 26 de junio de 2019)
- Christopher Olah(2014*). Neural Networks, Manifolds, and Topology.* Recuperado de: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/. (Última consulta 26 de junio de 2019)
- Francisco Javier García Polo (2012). *Aprendizaje por refuerzo para la toma de decisiones segura en dominios con espacios de estados y acciones*

continuos. Recuperado de: https://e-archivo.uc3m.es/bitstream/handle/10016/17321/tesis_francisco-javier_garcia_polo_2013.pdf. (Última consulta 26 de junio de 2019)

- Guillermo Julian (2014). *Las redes neuronales: qué son y por qué están volviendo.* Recuperado de: https://www.xataka.com/robotica-e-ia/las-redes-neuronales-que-son-y-por-que-estan-volviendo. (Última consulta 26 de junio de 2019)
- IBM Knowledge Center. *Redes neuronales.* Recuperado de: https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/neuralnet/idh_neuralnet_model_options.html#idh_neuralnet_model_options. (Última consulta 26 de junio de 2019)
- Luis Federico Bertona (2005). *Entrenamiento de redes neuronales basado en algoritmos evolutivo*. Recuperado de: http://laboratorios.fi.uba.ar/lsi/bertona-tesisingenieriainformatica.pdf. (Última consulta 26 de junio de 2019)
- María G. Longoni, Eduardo A. Porcel, María V. López, Gladys N. Dapozo (2010). *Modelos de Redes Neuronales Perceptrón Multicapa y de Base Radial para la predicción del rendimiento académico de alumnos universitario.* Recuperado de: http://sedici.unlp.edu.ar/bitstream/handle/10915/19333/070.pdf?sequence=1. (Última consulta 26 de junio de 2019)